(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(11) **DE 3628286 A1**

(51) Int. Cl. 4: G06F 15/62

(21) Aktenzeichen: P 36 28 286.3
(22) Anmeldetag: 20. 8. 86
(43) Offenlegungstag: 25. 2. 88

DE 3628286 A1

(71) Anmelder:
Stärk, Jürgen, Dipl.-Ing. Dipl.-Inform., 6072 Dreieich, DE

(72) Erfinder:
gleich Anmelder

(54) Prozessor mit integriertem Speicher

Herkömmliche Rasterdisplays haben einen separaten Graphikprozessor und Bildspeicher. Der Bildspeicher ist z. B. durch einen 8, 16 oder 32 Bit-Datenbus mit dem Graphikprozessor verbunden. Solche Anordnungen haben den prinzipiellen Nachteil, daß je nach Busbreite nur eine eingeschränkte Anzahl von Bildpunkten modifiziert werden kann.
Der Erfindung liegt die Aufgabe zugrunde, erweiterte Zugriffsmöglichkeiten auf den Inhalt des Bildspeichers zu schaffen.
Dies wird dadurch erreicht, daß der Prozessor oder seine datenmanipulierende Einheit und der Bildspeicher auf einem Chip integriert werden. Dadurch ist es möglich, Zeilen oder Spalten oder Teile davon oder Blöcke von Speicherzellen bzw. Bildpunkte zu adressieren und zu modifizieren.
Der Leistungszuwachs wird
- durch eine spezielle Struktur,
- durch den Aufbau des Bildspeichers, der von einem konventionellen Speicher weitgehend verschieden ist,
- durch die spezielle Adressierlogik und
- durch die stark erweiterte Datenbusbreite, die nur auf einem Chip möglich ist,
erreicht.
Der hauptsächliche Anwendungsbereich ist in der Rastergraphik zu finden, z. B. bei Mikrocomputern, Terminals oder anderen Raster-Scan-Konvertern.

DE 3628286 A1

Patentansprüche

1. Prozessor in Verbindung mit einem Bildspeicher, z. B. für ein Rasterdisplay, **dadurch gekennzeichnet,** daß der Prozessor ganz oder teilweise und der Bildspeicher auf demselben Chip integriert sind.
2. Prozessor nach Anspruch 1, dadurch gekennzeichnet, daß ein Siliziumchip verwendet wird.
3. Prozessor nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß die datenmanipulierende Einheit des Prozessors — über einen der Anzahl der Speicherzellen in einer Zeile und/oder Spalte des Bildspeicher entsprechend breiten Bus — mit dem Bildspeicher gekoppelt sein kann.
4. Prozessor nach Anspruch 1, 2 oder 3, dadurch gekennzeichnet, daß die datenmanipulierende Einheit des Prozessors aus einem Barrel-Shifter mit der Busbreite der datenmanipulierenden Einheit bestehen kann.
5. Bildspeicher nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß das Speicherfeld mit einer speziellen Verbindungslogik ausgestattet ist, die es ermöglicht, die Speicherinformationen, die in einer gedachten horizontalen Achse gespeichert sind, über eine vertikale Achse auszulesen und zu überschreiben. Ebenso ist es möglich, Informationen, die in einer gedachten vertikalen Achse gespeichert sind, über eine horizontale Achse auszulesen.
6. Bildspeicher nach Anspruch 1, 2 oder 5, dadurch gekennzeichnet, daß Zeilen oder Spalten von Speicherzellen — oder Teile davon — in einem Speicherzyklus gelesen oder beschrieben werden können.
7. Bildspeicher nach Anspruch 1, 2, 6 oder 7, dadurch gekennzeichnet, daß auch rechteckige Blökke von Speicherzellen in einem Zugriffszyklus überschrieben werden können.
9. Bildspeicher nach Anspruch 1, 2, 6, 7 oder 8, dadurch gekennzeichnet, daß der Speicherselektionsmechanismus aus einem oder mehreren M-aus-N Decodern bestehen kann.
10. Bildspeicher nach Anspruch 1, 2, 6, 7, 8 oder 9, dadurch gekennzeichnet, daß die vom Adressdekoder selektierten Adressleitungen mit Hilfe von z. B. einem Schieberegister verändert werden können.
11. Bildspeicher nach Anspruch 1, 2, 6, 7, 8, 9 oder 10, dadurch gekennzeichnet, daß Speicherbereiche mit Hilfe einer Logik, wie sie z. B. in **Fig.** 15 abgebildet ist, gesperrt werden können.
12. Bildspeicher nach Anspruch 1, 2, 6, 7, 8, 9, 10 oder 11, dadurch gekennzeichnet, daß der Auswahlmechanismus für die aufzufrischende (Refresh) Zeile von Speicherzellen aus einem Schieberegister bestehen kann.

Beschreibung

Prozessor in Verbindung mit einem Bildspeicher, z. B. für ein Rasterdisplay, dadurch gekennzeichnet, daß der Prozessor und der Bildwiederholspeicher auf demselben Chip integriert sind.

Die Erfindung betrifft einen Prozessor nach dem Oberbegriff des Anspruchs 1, insbesondere für graphische Anwendungen.

Der Bildspeicherinhalt in einem Rasterdisplaysystem soll schnell modifiziert werden. Dafür ist es häufig nötig, mehrere Bildpunkte gleichzeitig zu verändern. Hieraus resultiert ein schneller Bildaufbau.

Herkömmliche Rasterdisplays haben einen seperaten Graphikprozessor und Bildspeicher. Der Bildspeicher ist z. B. durch einen 8, 16 oder 32 Bit Datenbus mit dem Graphikprozessor verbunden. Solche Anordnungen haben den prinzipiellen Nachteil, daß je nach Busbreite nur eine eingeschränkte Anzahl von Bildpunkten in einem Graphikprozessorzyklus verändert werden können.

Der Erfindung liegt die Aufgabe zugrunde, erweiterte Zugriffs- und Modifikationsmöglichkeiten auf den Inhalt des Bildspeichers zu schaffen.

Diese Aufgabe wird bei einer gattungsgemäßen Einrichtung durch die kennzeichnenden Merkmale des Anspruches 1 gelöst.

Die mit der Erfindung erzielbaren Vorteile bestehen insbesondere darin, daß Zeilen und/oder Spalten von Bildpunkten, oder Teile davon, im Bildspeicher verschoben und/oder modifiziert werden können und darin, daß Zeilen, Spalten oder Blöcke von Bildpunkten überschrieben werden können. Insbesondere das Generieren von gefüllten oder schraffierten Objekten wie z. B. Rechteck, Parallelogramm, Trapez, Kreis, Polygon kann signifikant schneller als in einer Anlage nach dem Stand der Technik ausgeführt werden. Operationen wie Windowclipping, Scrolling und Bit-Block Transfer (BitBlt) sind erheblich schneller möglich. Die Erzeugung der Synchronisationssignale zur Bildröhrensteuerung kann mit wenig Aufwand integriert werden.

Die Erfindung wird anhand der Zeichnungen näher erläutert.

Fig. 1 zeigt ein konventionelles System.

Fig. 2 sind Prozessor und Bildspeicher durch einen expandierten Datenbus verbunden.

Fig. 3 zeigt das Grundprinzip der Erfindung.

Fig. 4 zeigt das stilisierte Speicherfeld mit Bezeichnung der Datenbusse und eingezeichneten Speicherzellen mit zwei Ausgangsbusse.

Fig. 5 zeigt eine vereinfachte Speicherzelle.

Fig. 6 zeigt das Speicherfeld mit den Verbindungen in der Diagonalen.

Fig. 7 zeigt den prinzipiellen Speicheraufbau.

Fig. 8 zeigt den Adresspfad.

Fig. 9 veranschaulicht die Selektierung einer Zeile von Speicherzellen.

Fig. 10 veranschaulicht die Selektierung einer Spalte von Speicherzellen.

Fig. 11 veranschaulicht die Selektierung eines Blokkes von Speicherzellen.

Fig. 12 zeigt die drei funktionalen Blöcke des Dekoders.

Fig. 13 zeigt die Adresstransformationseinheit.

Fig. 14 zeigt eine Zelle zum Sperren von Speicherbereichen.

Fig. 15 zeigt ein Beispiel für das Sperren von Speicherbereichen.

Fig. 16 zeigt die datenmanipulierende Einheit.

Eine Anlage gemäß dem Stand der Technik ist in **Fig.** 1 dargestellt. Pos. 1 zeigt einen Prozessor. Pos. 2 ist ein Sekundärspeicher. Pos. 3 ist eine Einheit zur Kommunikation des Systems mit der Außenwelt. Parallel dazu können weitere Baugruppen 4 geschaltet sein. Pos. 7 ist eine Bildschirmsteuereinheit, welche die Bildschirminformation aus dem Speicher 6 ausliest und durch Pos. 8, eine Bildröhre, darstellt. Die Teile **1, 2, 3, 4, 7** sind durch einen Datenbus 5 an einen gemeinsamen Speicher 6 gekoppelt. Bei dieser Anordnung ist der Prozessor 1 derjenige, der die Bildinformation im Speicher 6 verändert. Der Prozessor 1 und der Speicher 6 sind auf ver-

schiedenen Chips untergebracht. Der Prozessor muß über einen 8, 16 oder 32 Bit breiten Datenbus 5 die Information im Speicher 6 verändern. Durch Pos. 5 wird die Geschwindigkeit der Kommunikation zwischen den Baugruppen Speicher und Prozessor eingeschränkt.

Fig. 2 zeigt eine nach der Erfindung aufgebaute Anlage. Der Bildspeicher 9 ist mit dem Prozessor 10 über die dazwischen liegenden Verbindungsleitungen verbunden. Beide Komponenten 9 und 10 sind auf einem Chip integriert. Insbesondere wird ein Siliziumchip verwendet. Außerdem dann die Anzahl der Verbindungsleitungen im Gegensatz zu der Anordnung nach dem Stand der Technik wesentlich vergrößert sein. Die Anzahl der Leitungen ist durch die Größe des Speichers 9 begrenzt. Sie entspricht der Anzahl der Speicherzellen in horizontaler oder vertikaler Richtung im Speicher 9. Wird ein Teil des Speichers nicht in dieser Weise angeschlossen, so ist jedoch immer noch die Funktion für den restlichen Teil gewährleistet. Sind zusätzliche nicht Speicherzellen an den Bus gekoppelt, so ist immer noch die Funktion gewährleistet. Der Prozessor kann aus 2 Teilen bestehen: Eine Steuereinheit und eine datenmanipulierende Einheit. Die Steuereinheit soll hier nicht weiter betrachtet werden und kann auch außerhalb des Chips untergebracht sein. Die datenmanipulierende Einheit, im weiteren Datenpfad genannt, und der Speicher sind im weiteren erfindungsgemäß beschrieben.

Fig. 3 zeigt den erfindungsgemäßen, prinzipiellen Zusammenhang des Bildspeichers 14 mit den Adress- (Pos. 11 und 12), Refresh- (Pos. 32) und Datenpfaden (Pos. 13) und deren erfindungsgemäßem Aufbau. Pos. 15 und 23 können Adressdekoder zum Selektieren von Speicherbereichen in Pos. 14 sein. Pos. 17 und 25 können Einheiten zum Verändern und/oder Speichern der in 15 und 23 erzeugten Adressvektoren sein. Pos. 19 und 27 können Einheiten zum Sperren von Speicherbereichen sein. Pos. 20 und 29 können Einheiten zum Erzeugen von Speicherrefreshadressen sein. Pos. 20 und 29 können zusätzlich zum Multiplexen des Eingangsvektors aus Pos. 21 bzw. 28 vorhanden sein. Pos. 32 kann eine Einheit zum Unterstützen des Bildschirmrefreshs sein.

Pos. 39 kann eine Einheit zur Vermeidung von Buskonflikten in 36 sein. Pos. 40 kann ein Register zur Speicherung eines Ergebnisses sein. Pos. 41 kann ein Schiebemechanismus, wie es ein Barrel-Shifter erfüllt, sein. Pos. 42 kann eine logische Verarbeitungseinheit sein, die zwei Eingangsvektoren mit Hilfe logischer Operationen verknüpft. Pos. 43 kann ein Speicherregister sein, welches einen Vektor speichern kann und auch den Datenaustausch dieses Systems mit der Außenwelt tätigen kann. Der Ausgang von 15 kann über 16 mit dem Eingang von 17 verbunden sein. Der Ausgang von 17 kann über 18 mit dem Eingang von 19 verbunden sein. Der Ausgang von 19 kann über 20 mit dem Eingang für die Spaltenselktierung der Pos. 14 verbunden sein. Der Ausgang von Pos. 23 kann über 24 mit dem Eingang von 25 verbunden sein. Der Ausgang von 25 kann über 26 mit dem Eingang von 27 verbunden sein. Der Ausgang von 27 kann über 28 mit 29 verbunden sein. Der Ausgang von 29 kann über 30 mit dem Eingang für Zeilenselektierung von Pos. 14 verbunden sein. Pos. 36 kann mit Pos. 14, 39, 43 und 42 verbunden sein. Der Ausgang von 43 kann über 38 mit dem Eingang von 42 verbunden sein. Pos. 38 und 36 bilden die Dateneingänge für Pos. 42. Diese Dateneingänge können die zu verarbeitenden Vektoren nach 42 übertragen. Der Ausgang von 42 kann über 37 mit dem Eingang von 41 gekoppelt sein. Der Ausgang von 41 kann über 35 mit dem Eingang von 40 gekoppelt sein. Der Ausgang von 40 kann über 34 mit dem Eingang von 39 gekoppelt sein. Verbindungen zum Steuern der Komponenten sind nicht eingezeichnet.

Pos. 32 kann eine Einheit zum Unterstützen des Bildschirmrefreshs sein. Ein Ausgang von Pos. 14 kann über 31 mit dem Eingang von Pos. 32 verbunden sein.

Der Bildspeicher (14) befindet sich in der Mitte des Bildes (Block Bildspeicher). Die Adresspfade 11 und 12 sind oben und links, der Datenpfad 13 rechts und der Refreshpfad 32 unten mit dem Speicher verbunden. Alle verbindenden Busse 16, 18, 21, 22, 24, 26, 28, 30, 31, 34, 35, 36, 37, 38 können als Verbindungsbreite die Anzahl der Zellen in horizontaler oder vertikaler Richtung im Bildspeicher haben.

Der Bildspeicher kann aus einem quadratischen Feld von Speicherzellen bestehen. Jede Speicherzelle ermöglicht das Speichern eines Bildpunktes. Das Feld kann in zwei Teile aufgeteilt werden. Teil *A* kann auf dem Bildschirm dargestellt. In Teil *B* ist es möglich, z. B. Zeichensätze oder Schraffurtypen zu speichern. Die Aufteilung des Feldes in Teil *A* und Teil *B* kann beliebig gewählt werden, da das Feld einen homogenen Aufbau hat. Die Aufteilung kann bei der Konstruktion des Chips festgelegt werden.

Erfindungsgemäß können durch 12 und 11 Adressen erzeugt werden, die in 14 einen Speicherbereich, z. B. eine Zeile oder eine Spalte von Speicherzellen, oder Teile davon, oder rechteckige Blöcke von Speicherzellen, adressieren. In dem adressierten Bereich können über 36 Informationen eingeschrieben oder ausgelesen werden. Werden Informationen ausgelesen, so können diese über 36 an 42 und 43 geführt werden. In 43 kann nun diese Information gespeichert werden. In 42 kann diese Information mit einer Information aus 43 über 38 logisch verknüpft werden und über 37 in 41 übertragen werden. In 41 kann nun diese Information zyklisch bitweise verschoben werden. Diese eventuell verschobene Information kann über 35 nach 40 transferiert werden und dort zwischengespeichert werden. Wird nun von 14 die Information nicht weiter auf 36 ausgegeben, so kann 39 über 36 die neu erzeugte Information in 14 oder in 43 abspeichern.

Eine Information, die aus 14 ausgelesen wird, kann erfindungsgemäß auch über 31 nach 32 transferiert werden. Dies ist in einem Speicherrefreshzyklus der Fall. Dann geben 20 über 22 nach 14 und 29 über 30 nach 14 entsprechende Adressen aus, und 14 kann über 31 nach 32 Bildschirminformation transferieren.

Fig. 4 zeigt erfindungsgemäß den prinzipiellen Aufbau der Speicherzellen und der Datenbusse des Speichers. Pos. 44 und Pos. 45 sind Datenbusse, über die die Kommunikation von und zu dem Speicher realisiert ist. An jeden Bus sind in diesem prinzipiellen Falle je 8 Datenleitungen (Pos. 46, 47) angeschlossen. Pos. 48 zeigt eine Speicherzelle. Sie ist oben und links mit den Datenleitungen der Datenbusse 45 und 46 verbunden. Dadurch ist es möglich, Zeilen von Speicherzellen durch den Bus *"A"* (44) und Spalten von Speicherzellen durch den Bus *"B"* (45) auszulesen und/oder zu beschreiben.

Fig. 5 zeigt eine vereinfachte statische dual-port Speicherzelle, um die Arbeitsweise des Speichers zu verdeutlichen. Pos. 50, 52, 54 und 59 sind Transistoren. 55 kann durch die Steuerleitung 54 der Zeilenauswahl dienen. 52 kann durch die Steuerleitung 54*a* der Spaltenauswahl der Speicherinformation Ein-/Ausgabe über 53 dienen. 50 kann durch die Steuerleitung 51 der Spaltenauswahl der Speicherinformation Ein/Ausgabe über 49 dienen. Die Speicherinformation ist in dem Ring, beste-

hend aus den zwei Invertern 56 und 57 und dem Transistor 59 gespeichert. Die Information kann ausgelesen werden, wenn 52, 55 und 59 geöffnet sind oder wenn 50, 55 und 59 geöffnet sind. Die Information kann überschrieben werden, wenn 59 geschlossen und 52 und 55 geöffnet sind oder wenn 59 geschlossen und 50 und 55 geöffnet sind. Pos. 56, 57 und 59 können durch einen Kondensator ersetzt werden. Dann ist die Steuerleitung von 59 und die Spannungsversorgungsleitungen der Zelle hinfällig.

Erfindungsgemäß können in allen Speicherzellen aus Fig. 4 die Transistoren 50 über 49 mit der an die jeweilige Zelle heranführende Datenbusleitung des Busses 45 verbunden sein. Ebenso können alle Transistoren 52 über 53 mit der an die jeweilige Zelle heranführende Datenbusleitung des Busses 44 verbunden sein.

Dies ermöglicht Lese- und Schreibtransfer von Daten über die beiden Busse 44 und 45. Der Transistor 55 ermöglicht die Zeilenselektierung, der Transistor 50 ermöglicht den Datenaustausch mit dem Bus *"B"* und der Transistor 52 den Datenaustausch mit dem Bus *"A"*. Im Speicherfeld sind alle Transistoren 52 einer Spalte von Speicherzellen und alle Transistoren 50 einer Zeile von Speicherzellen über die Busse miteinander verbunden.

Um nun auch Spalten von Speicherzellen durch den Bus *"A"* und Zeilen von Speicherzellen durch den Bus *"B"* auszulesen und/oder zu beschreiben, können zusätzliche Verbindungen angebracht werden. Fig. 6 zeigt die für die Funktion entscheidenden Verbindungen 60, 61, 62, 63, 64, 65, 66 und 67 in der Diagonalen des Speicherfeldes. Ansonsten entspricht Fig. 6 Fig. 4. Durch das Anbringen dieser Verbindungen werden jeweils die Datenleitung des Busses *"A"* und des Busses *"B"* verbunden, d. h.

Bus *"A"* Leitung 0 wird mit Bus *"B"* Leitung 0 und
Bus *"A"* Leitung 1 wird mit Bus *"B"* Leitung 1 etc. verbunden.

Das Lesen und Schreiben einer Zeile von Speicherzellen über den Bus *"A"* und das Lesen und Schreiben einer Spalte von Speicherzellen über den Bus *"B"* ist jetzt möglich. Soll z. B. ein Speicherwort über den Bus *"A"* in die erste Zeile eingeschrieben werden, so kann die Information, die auf dem Bus anliegt, über die Transistoren 52 und 55 in die jeweiligen Zellen eingelesen werden. Soll die linke Spalte von Speicherzellen über den Bus *"A"* überschrieben werden, so wird die Datenverbindung über den Transistor 50 und 55 in der jeweiligen Speicherzelle gewählt. Das zweite Bit der Spalte wird auf folgende Weise gesetzt oder gelöscht:

Die Information auf dem zweiten Bit des *A* Busses 44 wird an der Verbindungsstelle 61 auf den *B* Bus 45 übergeführt. Vom *B* Bus aus kann die Information über den Transistor 50 und 55 der zweiten Zelle in die Zelle geladen werden. Bei all diesen Ausführungen wird vorausgesetzt, daß der zu lesende oder zu modifizierende Bereich adressiert ist.

Fig. 7 zeigt erfindungsgemäß prinzipiell außer den Datenbussen auch die Adressleitungen und Steuerleitungen mit der entsprechenden zusätzlichen Hardware. Die Hardware kann zur Generierung zusätzlicher Signale dienen. Pos. 69 und 70 sind die schon beschriebenen Busse. Pos. 71 ist ein Inverter und dient zur Invertierung des Signals 92. Pos. 72 ist die Zeilenadresse. Sie kann aus einem binären Vektor bestehen. Pos. 68 ist die Spaltenadresse. Sie kann aus einem binären Vektor bestehen. Pos. 73 ist die optionale Freischaltung der Zeilenadresse. Pos. 74 ist das Symbol für ein "And"-Baustein. Pos. 75 kann ein Speicherfreigabesignal sein. Dieses Signal kann in allen Zellen mit Pos. 58 verbunden sein. Pos. 76 kann mit der Pos. 54a aller Speicherzellen in dieser Spalte verbunden sein. Pos. 77 kann mit allen Pos. 51 der Speicherzellen in dieser Spalte verbunden sein. Das gleiche wie für Pos. 76 gilt für die jeweilige Spalte von Speicherzellen für 78, 80, 82, 84, 86, 88 und 90. Das gleiche gilt analog wie für Pos. 77 für Pos. 79, 81, 83, 85, 87, 89 und 91. Pos. 92 gibt an, ob eine Spalte oder eine Zeile adressiert werden soll. Erfindungsgemäß kann das Einschreiben einer Information in den Speicher folgenden Ablauf haben: Zunächst können die Adressen an 68 und 72 angelegt werden. 92 kann eingestellt werden. Dadurch kann eine der Selektionsleitungen 76 bis 91 freigegeben werden. Eine Information kann auf einem der Busse 45 oder 44 angelegt werden. Das Signal 75 kann gelöscht werden. Das Signal 73 kann gegeben werden. Nun wird die Information eingeschrieben. Sind in 72 und 68 Bereiche adressiert worden, so kann nun ein rechteckiges Feld von Speicherzellen überschrieben werden. Wird auf den Bussen 45 oder 44 keine Information angelegt, so können bei gehaltenem Signal 75 Informationen ausgelesen werden. Im Gesamtsystem kann der Bus 31 dem Bus 70, der Bus 36 dem Bus 69, der Bus 30 dem Bus 72 und der Bus 22 dem Bus 68 entsprechen.

Fig. 8 zeigt den Adresspfad 12 oder 11, der zum Generieren von Adressen für den Bildspeicher vorgesehen ist. Pos. 93 kann ein Adresskoder (1-aus-N) oder ein spezieller Dekoder (M-aus-N) sein. Der Ausgang von Pos. 93 kann über 94 mit dem Eingang von 95 gekoppelt sein. In 95 können die von 93 generierten Signale gespeichert oder modifiziert werden. Der Ausgang von 95 ist über 96 mit dem Eingang von 97 gekoppelt. Hier kann ein bestimmter Speicherbereich gegen Überschreiben geschützt werden. Der Ausgang von 97 ist über 98 mit dem Eingang von 99 verbunden. Pos. 99 kann zur Unterstützung des Bildschirm- und des Speicherrefreshs dienen. Pos. 100 ist der Ausgang des Adresspfades.

Erfindungsgemäß kann ein Adresspfad 11 oder 12 verwendet werden, um die Zeilen- bzw. die Spaltenadresse zu generieren. Um erweiterte Adressierungsmöglichkeiten zu erreichen, kann ein Adresspfad aus einem Pos. 93, einer Pos. 95, einer Pos. 97 und einer Pos. 99 für den Refresh dynamischer Speicherzellen bestehen. In dem Adressdekoder 93 wird zunächst ein Adressbereich für den Bildspeicher aus kodierten Eingangsadressen erzeugt. Dieser Adressbereich kann in 95 und in 97 gespeichert werden oder direkt als Adresse dem Bildspeicher zugeführt werden. In 95 kann der gespeicherte Adressvektor verschoben oder mit einem schon gespeicherten Vektor verknüpft werden. Dies ermöglicht sehr schnelle und variationsreiche Adressierungsmöglichkeiten. In der Adresstransformationseinheit kann zwischen internem und Dekodervektor als Ausgabevektor gewählt werden. Der somit erzeugte Adressvektor wird in 97 logisch AND mit dem dort gespeicherten Vektor verknüpft. Dadurch ist ein Sperren von Speicherbereichen möglich. Der nun entstandene Vektor wird in dem normalen Arbeitsmodus dem Bildspeicher als Ansteuervektor für die Adressen übergeben. In dem Refreshzyklus wird statt dieses Vektors der Refreshvektor angelegt. Dies geschieht im letzten Block des Adresspfades.

Erfindungsgemäß können zwei (M-aus-N) Dekoder 93, je einer für die Zeilen- und Spaltenauswahl, einen Bereich von Speicherzellen auswählen, die adressiert werden. Jeder dieser Adressdekoder kann mit einer größten und einer kleinsten gewünschten Adresse ange-

steuert werden. Der Dekoder selektiert dann alle Speicherzeilen oder -spalten, die zwischen diesen beiden Adressen liegen. Durch die beiden Dekoder können nun Zeilen, Spalten oder Blöcke von Speicherzellen in der Speichermatrix adressiert werden.

Fig. 9 veranschaulicht die Selektierung einer Zeile von Speicherzellen. Pos. 101 und 102 können z. B. zwei Adressdekoder oder Adresspfade sein. Pos. 103 zeigt den Speicher. Pos. 104 zeigt die ausgewählte Zeile im Speicher. Pos. 105 zeigt die letzte ausgewählte Spalte und Pos. 106 die erste ausgewählte Spalte im Speicher. Pos. 106b zeigt den adressierten Speicherbereich, der aus mehreren Zellen in einer Zeile bestehen kann.

Fig. 10 veranschaulicht die Selektierung einer Spalte von Speicherzellen. Die Bezeichnung der Baugruppen kann aus Fig. 9 entnommen werden. Pos. 107 zeigt die letzte adressierte Zeile und Pos. 108 die erste adressierte Zeile im Speicher. Pos. 109 zeigt die adressierte Spalte im Speicher. Pos. 110 zeigt den adressierten Speicherbereich, der aus mehreren Zellen bestehen kann.

Fig. 11 veranschaulicht die Selektierung eines Blokkes von Speicherzellen. Die Bezeichnungen der Baugruppen ist aus Fig. 7 zu entnehmen. Pos. 111 bezeichnet die erste selektierte Zeile und Pos. 112 die letzte selektierte Zeile von Speicherzellen. Pos. 113 zeigt die erste und Pos. 114 die letzte selektierte Spalte von Speicherzellen. Pos. 115 zeigt den Bereich von Speicherzellen, die adressiert sind.

Fig. 12 zeigt drei funktionale Blöcke des Dekoders. Pos. 117 zeigt einen Dekoder, der ab einer gegebenen Eingangsadresse 119 alle Selektionsleitungen freigibt. Die Eingangsadresse 119 kann kodiert sein, die Selektionsleitungen 122 nicht. Pos. 118 zeigt einen Dekoder, der von der Adresse 0 bis zu einer gegebenen Eingabeadresse 120 alle Selektionsleitungen 121 freigibt. Die Eingangsadresse 120 kann kodiert sein. Die Ausgangsadresse nicht. Folgendes Beispiel sei dazu genannt:

Die Busse 117 und 120 haben eine Breite von 9 Bit, die Busse 122 und 121 haben eine Breite von 512 Bit.

Die Ausgänge von 117 und 118 sind über 121 und 122 mit den Eingängen von 116 verbunden. Hier können die beiden Eingangsbusse bitweise logisch "AND" miteinander verknüpft werden. Die Gesamtfunktion, die sich daraus ergibt, ist folgende:

Durch Anlegen einer Minimumadresse an 117 und durch Anlegen der Maximumadresse an 118 wird an 123 ein Bereich von Leitungen des Busses selektiert. Die Busbreite von 123 ist gleich der Busbreite von 121 oder 122.

Fig. 13 zeigt die Adresstransformationslogik, welche dem schnellen Generieren, Speichern und Modifizieren von Adressen und Adressbereichen dient. Pos. 125 zeigt ein Schieberegister, welches parallel ladbar und nachladbar ist. Das heißt, der gespeicherte Vektor im Schieberegister kann logisch "OR" mit einem an 124 anstehenden Vektor verknüpft und gespeichert werden. Außerdem ist es möglich, mit diesen modifizierten Schieberegistern bei dem Schiebevorgang ein logisches "OR" oder "AND" zwischen dem gespeicherten Vektor und dem um ein Bit verschobenen Vektor vorzunehmen. Für das logische "OR" bedeutet das:

Aus dem Vektor 0001100000
wird 0001110000
bei einer Verschiebung nach rechts und 00111100000
bei einer Verschiebung nach links. Für das logische "AND" bedeutet das:

Aus dem Vektor 0000111100 wird 0000011100
bei einer Verschiebung nach rechts
und 0000111000
bei einer Verschiebung nach links.

Pos. 127 ist ein Multiplexer, der die Busse 124 und 126 multiplexen kann. Pos. 129 ist das gleiche Schieberegister wie Pos. 125. Pos. 131 ist eine logische Einheit, mit der es möglich ist, die zwei Busse 128 und 130 mit Hilfe logischer Operationen zu verknüpfen. Pos. 124 ist der Eingangsbus in 125 und 127 . Der Ausgang von 125 ist über 126 mit dem Eingang von 127 gekoppelt. Der Ausgang von 127 ist mit dem Eingang von 129 über 128 und mit dem Eingang von 131 gekoppelt.

Ein Bitvektor kann über 124 in die Schieberegister geladen werden. Da der Adressdekoder, der die Eingangsadressen erzeugt, nur zusammenhängende Adressbereiche generieren kann, sind mit Hilfe von 125 oder 129 erweiterte Adressierungsarten erschlossen. Es ist somit möglich, beliebig viele nicht zusammenhängende Bereiche des Bildspeichers zu adressieren. Anwendungen dafür sind im gleichzeitigen Zeichnen gleicher Objekte zu finden. Nach dem ersten Schieberegister kann sich ein Multiplexer befinden. Mit ihm ist es möglich, den Adressvektor aus dem Schieberegister oder dem Dekoder auszuwählen. Der so ermittelte Vektor kann in das zweite Schieberegister eingelesen werden. In dem nachfolgenden Block können die Adressvektoren des zweiten Schieberegisters und der bis hierhin selektierte Adressvektor mit Hilfe der 16 möglichen logischen Operationen verknüpft werden. Diese Kette von Schieberegistern, Multiplexen und logischen Einheiten kann beliebig erweitert werden. Die Pos. 124, 126, 128, 130 und 132 können die gleiche Busbreite wie die Ausgänge der Adressdekoder haben. Die Pos. 125, 127, 129 und 131 haben die Verarbeitungsbreite wie die dazwischen liegenden Busse.

Fig. 14 zeigt erfindungsgemäß eine Ein-Bit-Zelle zum Sperren von Speicherbereichen. Pos. 136 ist ein Latch. Pos. 133 ist eine Eingangsbitleitung. Pos. 135 verknüpft den Ausgang von 136 mit 133 zu 134 mit der logischen Operation "AND". Beim Setzen des Speicherbereiches, der vor dem Überschreiben geschützt sein soll, wird in jede dieser Zellen, die in dem Adresspfad liegen, eine Null für das Sperren der jeweiligen Zeile oder Spalte und eine Eins für das Freigeben der jeweiligen Zeile oder Spalte eingeschrieben. In dem Adresspfad muß für jede Adressleitung solch eine Zelle vorhanden sein, um Speicherbereiche vor dem Überschreiben zu schützen. Dies kann in der Clipping-Einheit 19 und/oder 27 erfolgen. Hierbei wird der Adressvektor mit einem Vektor, der in diesem Block gespeichert ist, logisch "AND" verknüpft, woraus sich der neue Adressvektor ergibt. Wird dieses Verfahren für den Adresspfad der Zeilenadressen und für den Adresspfad der Spaltenadressen angewendet, so kann z. B. ein rechteckiges Fenster im Bildspeicher beschrieben werden. Der restliche Bereich des Bildspeichers ist dann gesperrt.

Fig. 15 zeigt ein Beispiel. Pos. 137 und 138 können M-aus-N Dekoder sein. Pos. 139 kann der Bildspeicher sein. Pos. 144 und 145 kann die Clipping-Einheit für die Zeilen- und Spaltenselektierung sein. Pos. 140 kann die kleinste adressierbare Spalte im Fenster sein. Pos. 141 kann die größte adressierbare Spalte im adressierbaren Fenster sein. Pos. 148 kann die kleinste adressierte Spalte im Speicher sein. Pos. 149 kann die größte adressierte Spalte im Speicher sein. Pos. 150 kann die größte adressierbare Zeile im Fenster sein. Pos. 153 kann die kleinste adressierbare Zeile im Fenster sein. Pos. 151 kann die größte adressierte Zeile im Speicher sein. Pos. 152 kann

die kleinste adressierte Zeile im Speicher sein. Pos. 142, 143, 146 und 147 begrenzen den Bereich im Speicher, der nun adressiert wurde. Das Fenster, das in 144 und 145 definiert wurde, war kleiner als der in 137 und 138 adressierte Bereich. Durch das Sperren von bestimmten Bereichen konnte wie in diesem Beispiel beim Schreiben eines Rechtecks von Speicherzellen ein Clippingvorgang getätigt werden, der nicht auf mathematischen Berechnungen beruht. Das Clippen kostet somit nur die Verzögerungszeit in 135.

In einem dynamischen Halbleiterspeicherbaustein müssen alle Speicherzelleninhalte während einer vom Herstellerprozeß abhängigen Zeit aufgefrischt ("refreshed") werden. Andernfalls geht die Information der Speicherzelle verloren. Im gezeigten Konzept werden alle Zeiten von Speicherzellen sequentiell aufgefrischt. In einem Refreshzyklus geschieht die Spaltenselektierung in 20. Der Adressdekoder wird abgeschaltet, und alle Spalten werden selektiert. Die Zeilenselektierung nutzt ein dynamisches Schieberegister in 29, in dem nach einer Resetphase nur ein Bit gesetzt wird. Die Verarbeitungsbreite von 20 und 29 entspricht der Breite von deren Ein- und Ausgangsbussen. Wenn die Adressen stabil sind, kann der Speicher zum Lesen freigegeben werden. Dadurch wird die adressierte Zeile von Speicherzellen aufgefrischt. Der Bildschirmrefresh kann vom Speicherrefresh abgeleitet sein. In jedem Bildschirmrefreshzyklus kann das Videoregister in dem Block 32 mit dem Teil der Daten gefüllt, der auf dem Bildschirm dargestellt werden soll. Der Speicherbereich, der nicht dargestellt wird, kann für z. B. einen Zeichensatz reserviert werden. Dieser Teil kann mit Hilfe von Blocktransferoperationen auf den sichtbaren Teil des Speichers übertragen werden.

Der Inhalt von 32 kann in Teilen von jeweils z. B. acht Bit ausgegeben werden. Extern werden diese Teile mit Hilfe eines schnellen Schieberegisters in einen seriellen Bitstrom umgewandelt. Dieser Bitstrom steuert den Bildschirm.

Fig. 16 zeigt den Datenpfad. Er wird für Transformation und Justierung von Bilddaten aus dem Bildspeicher, sowie zur Kommunikation mit der Außenwelt verwendet. Die Busbreite entspricht der Anzahl der Speicherzellen in einer Zeile oder einer Spalte des Bildspeichers. Dadurch können sehr lange Vektoren vom Bildpunkt in einem internen Zyklus verändert werden. Pos. 164 stellt die Datenverbindung des Datenpfades und des Bildspeichers zur Außenwelt und das Speichermedium für Bildpunktevektoren im Datenpfad dar. Ein Vektor kann entweder aus dem Bildspeicher oder wortweise über den Data-Bus geladen werden. Der in 163 gespeicherte Vektor kann in der Einheit 161 bitweise mit einem Vektor, der aus dem Bildspeicher ausgelesen wird, verknüpft werden. Alle 16 logischen Operationen zwischen den zwei Vektoren sind möglich. Der durch die Operation entstandene neue Vektor kann, z. B. um BitBlt (Bit-Block-Transfer) zu ermöglichen, durch den Barrel-Shifter beliebig rotiert werden. Ein dem Shifter nachgeschaltetes Register dient der Steuerung des Datenpfades. Der nachgeschaltete Tristate-Buffer dient der Vermeidung von Buskonflikten im I/O-Bus.

Nummer: 36 28 286
Int. Cl.4: G 06 F 15/62
Anmeldetag: 20. August 1986
Offenlegungstag: 25. Februar 1988

Fig. 1

3628286

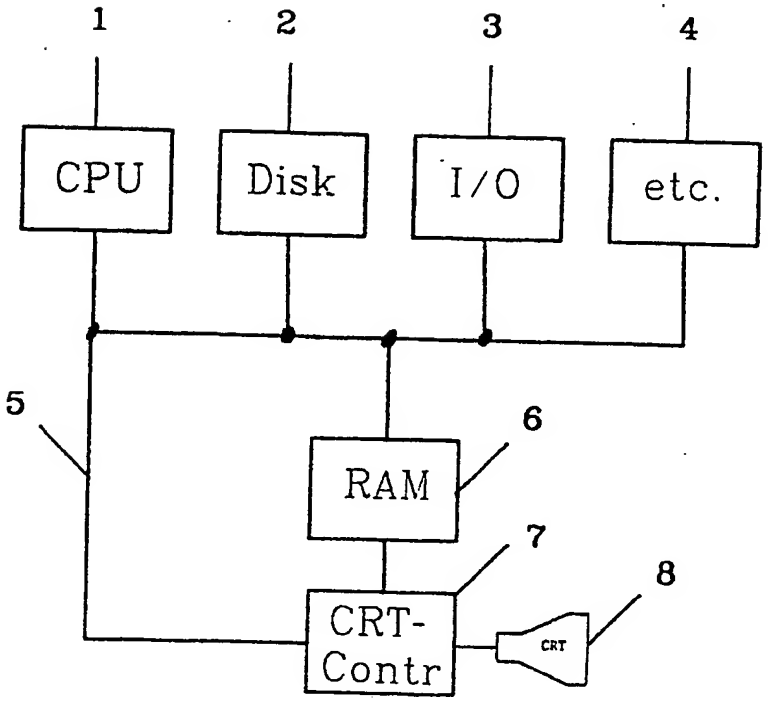


Fig. 2

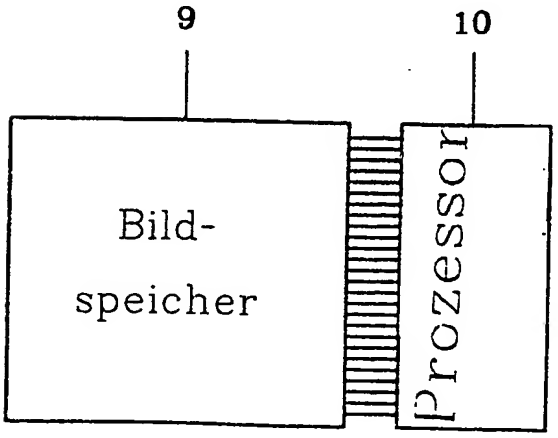


Fig. 3

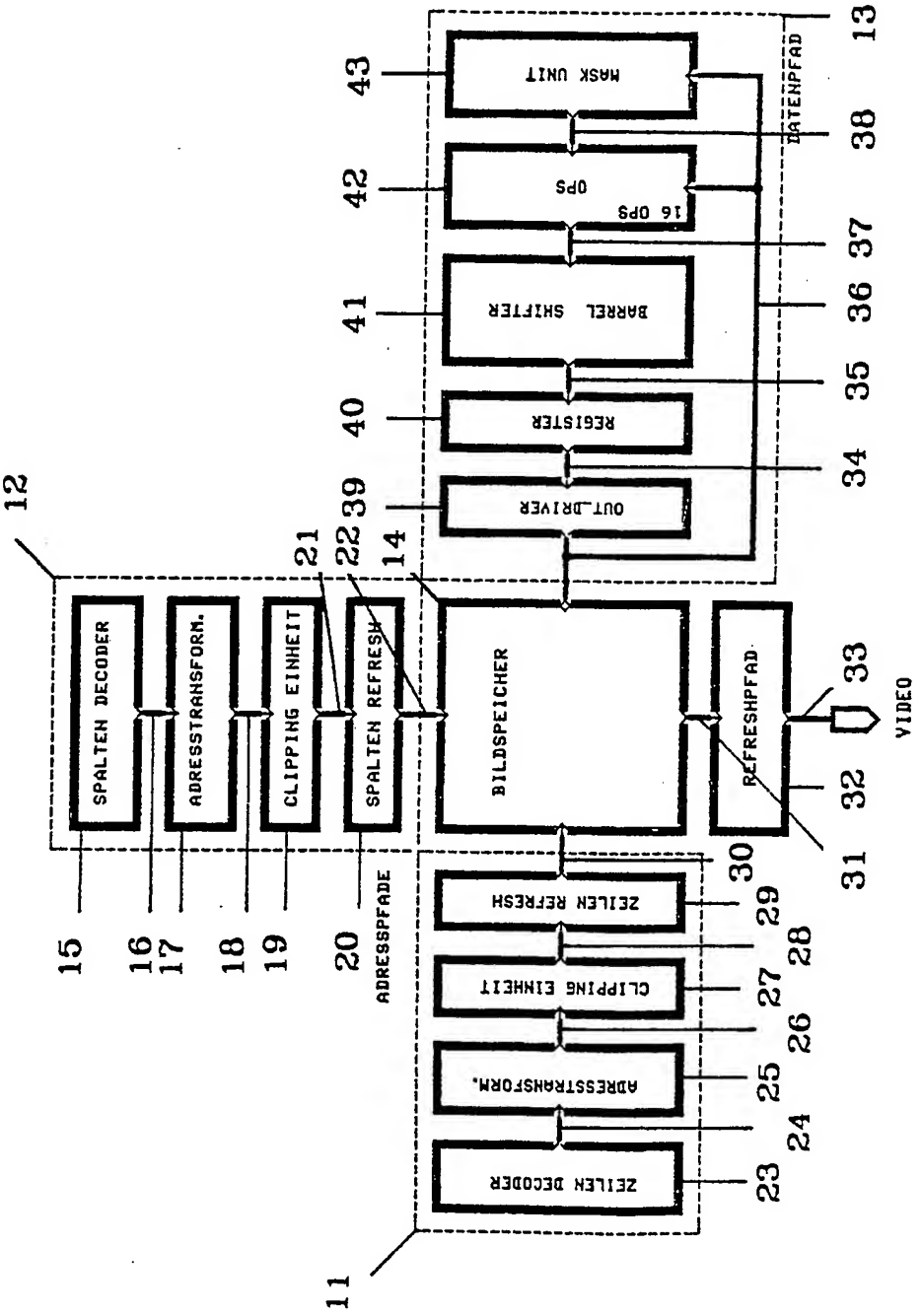
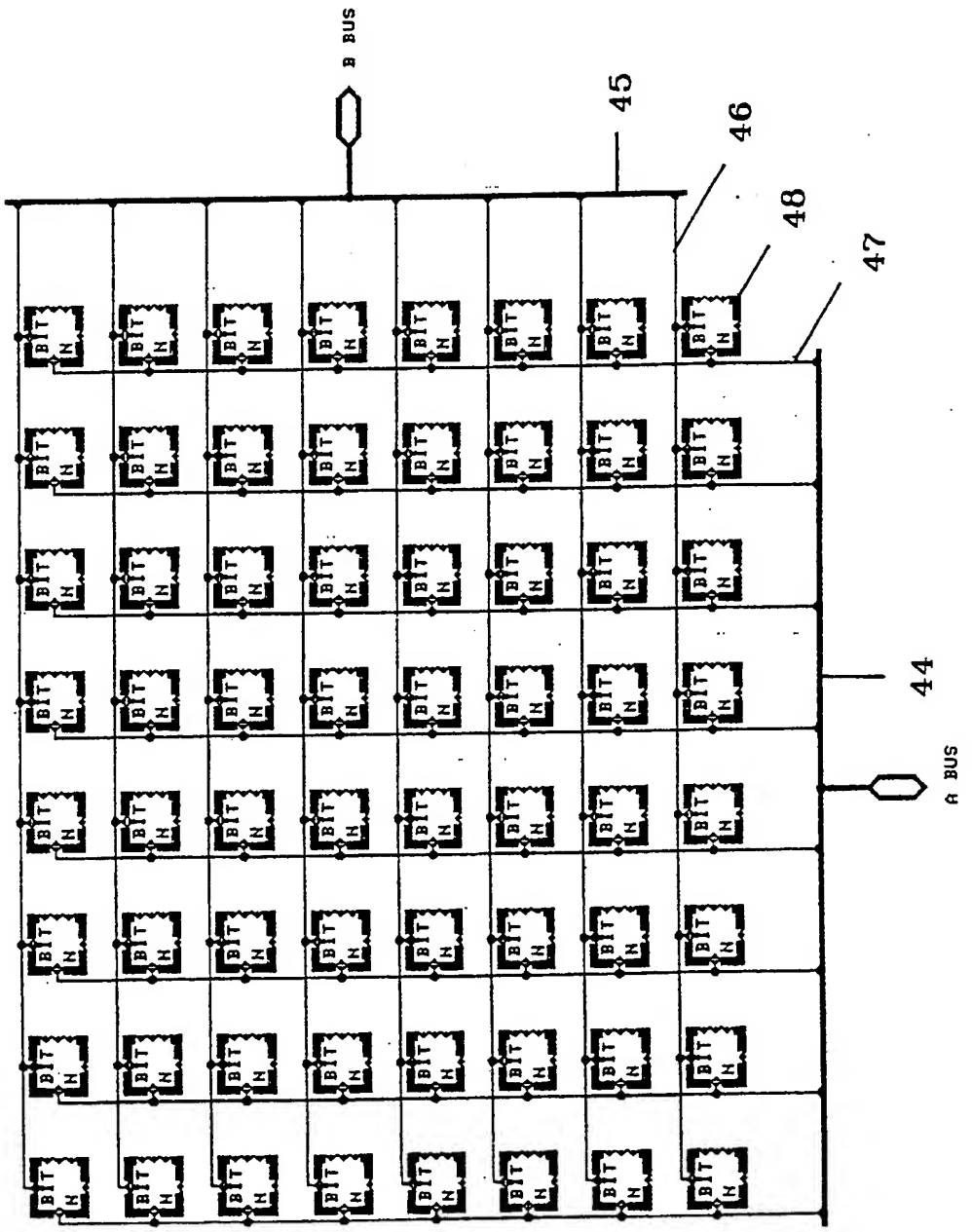
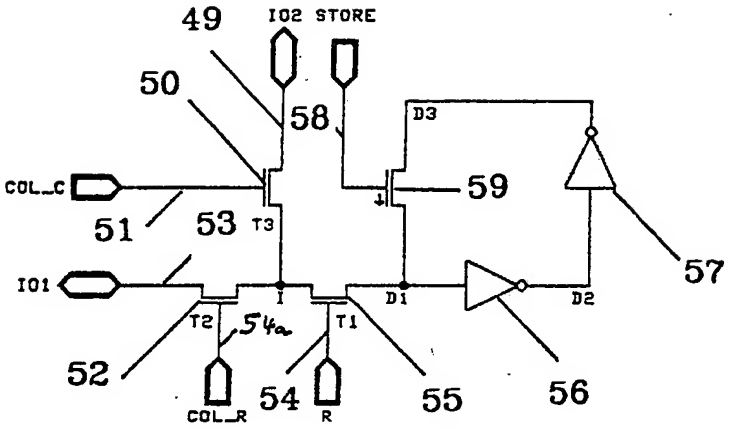

MASK UNIT
OPS
16 OPS
BARREL SHIFTER
REGISTER
OUT_DRIVER
DATENPFAD
SPALTEN DECODER
ADRESSTRANSFORM.
CLIPPING EINHEIT
SPALTEN REFRESH
BILDSPEICHER
REFRESHPFAD
VIDEO
ZEILEN REFRESH
CLIPPING EINHEIT
ADRESSTRANSFORM.
ZEILEN DECODER
ADRESSPFADE
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

Fig. 4



3628286

Fig. 5



3628286

Fig. 6

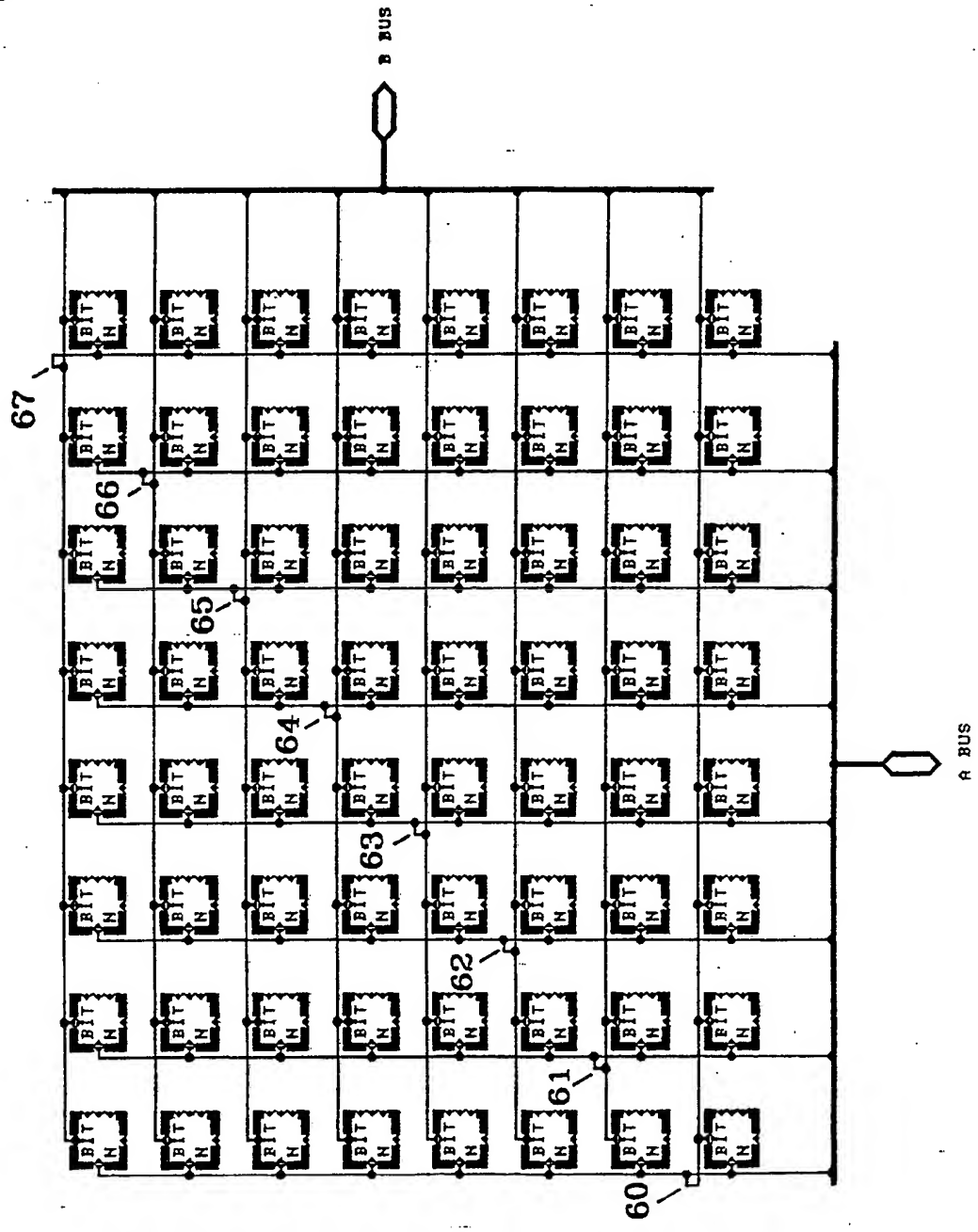


Fig. 7

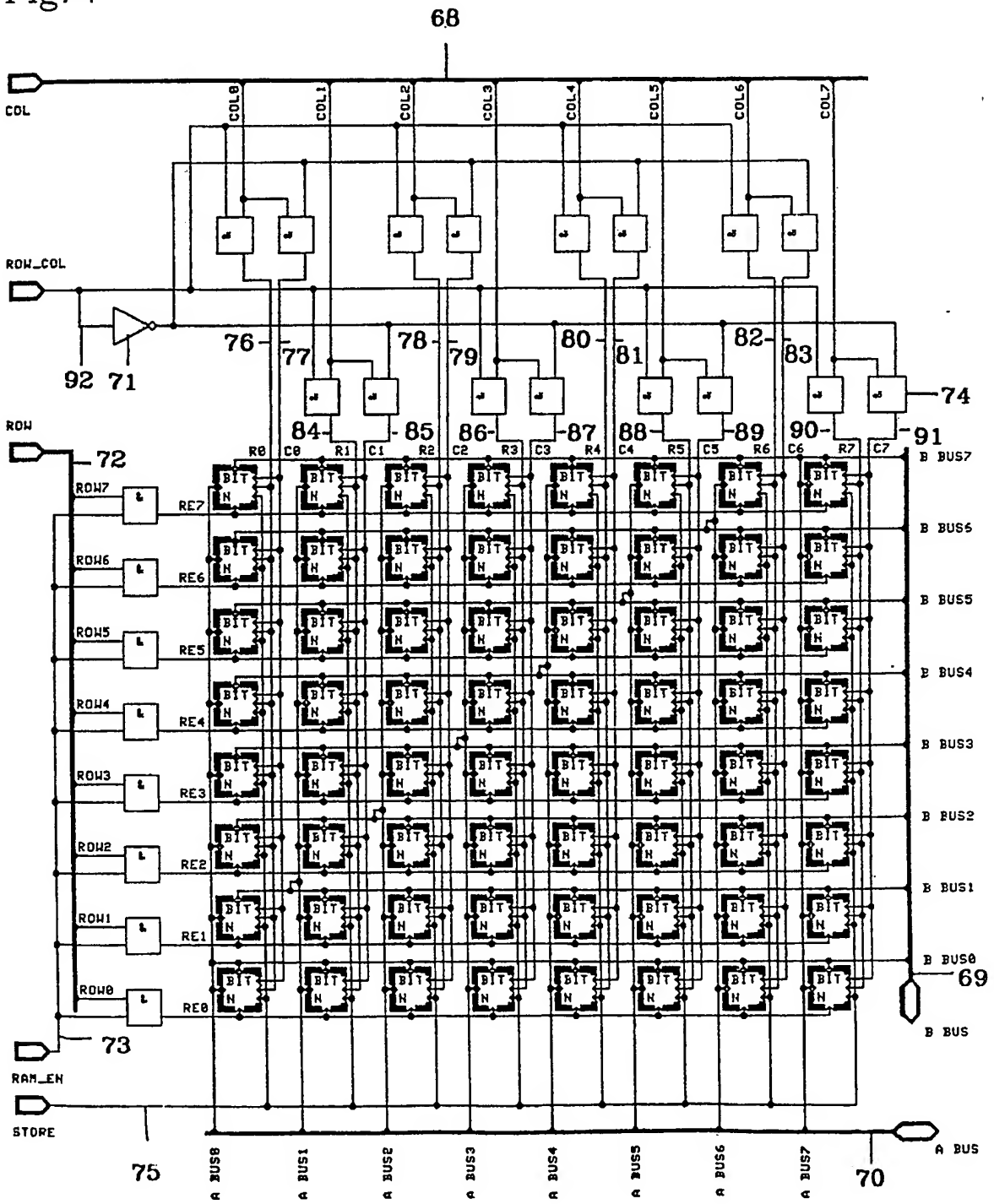


Fig. 8

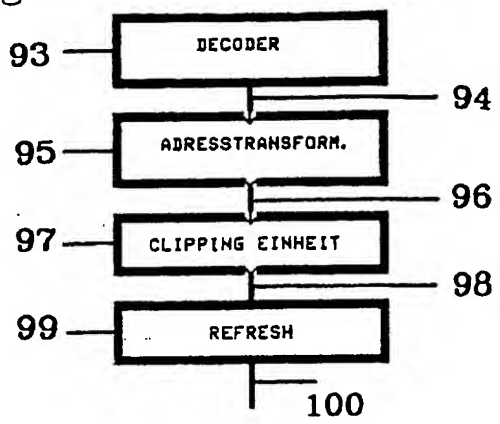
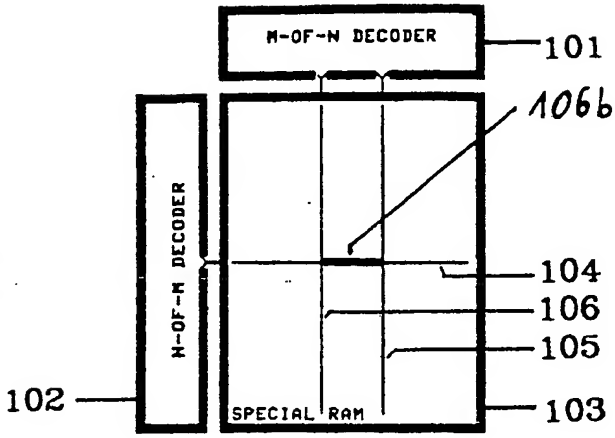


Fig. 9



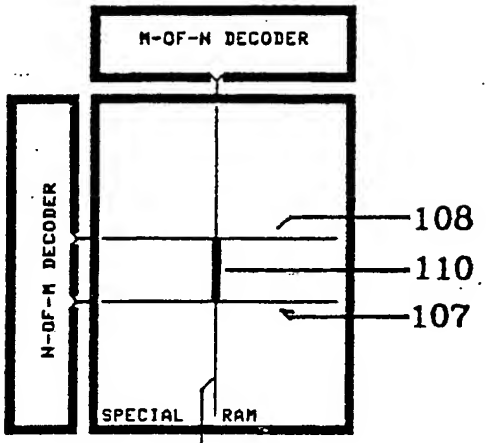


Fig. 10

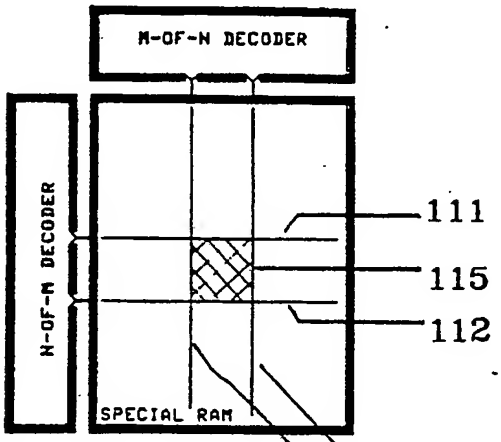


Fig. 11

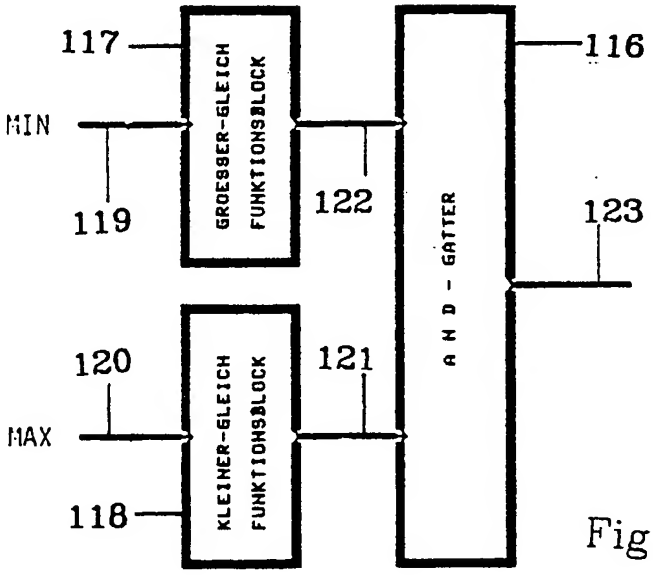
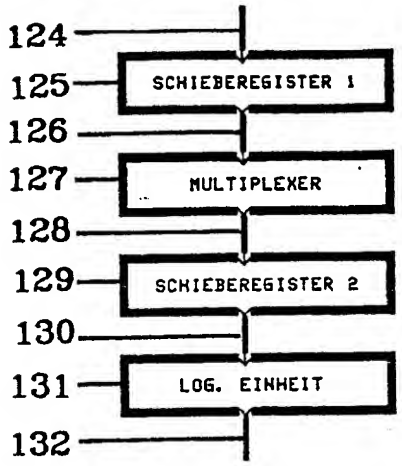


Fig. 12

Fig. 13



3628286

Fig. 14

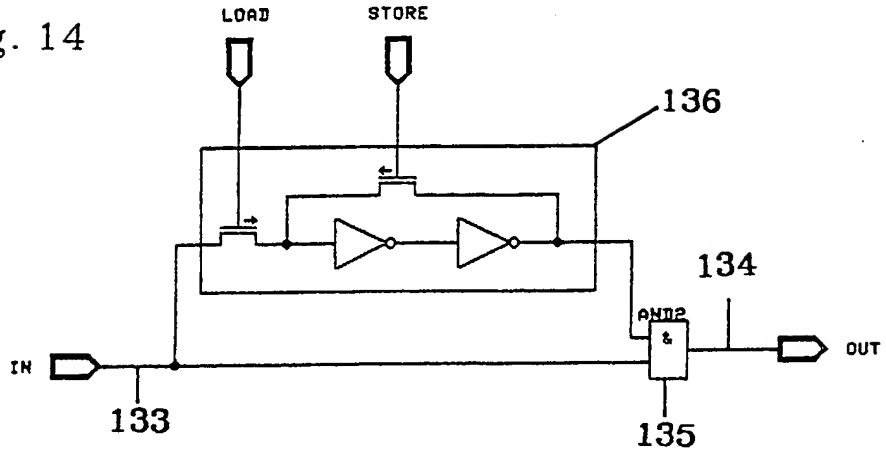


Fig. 15

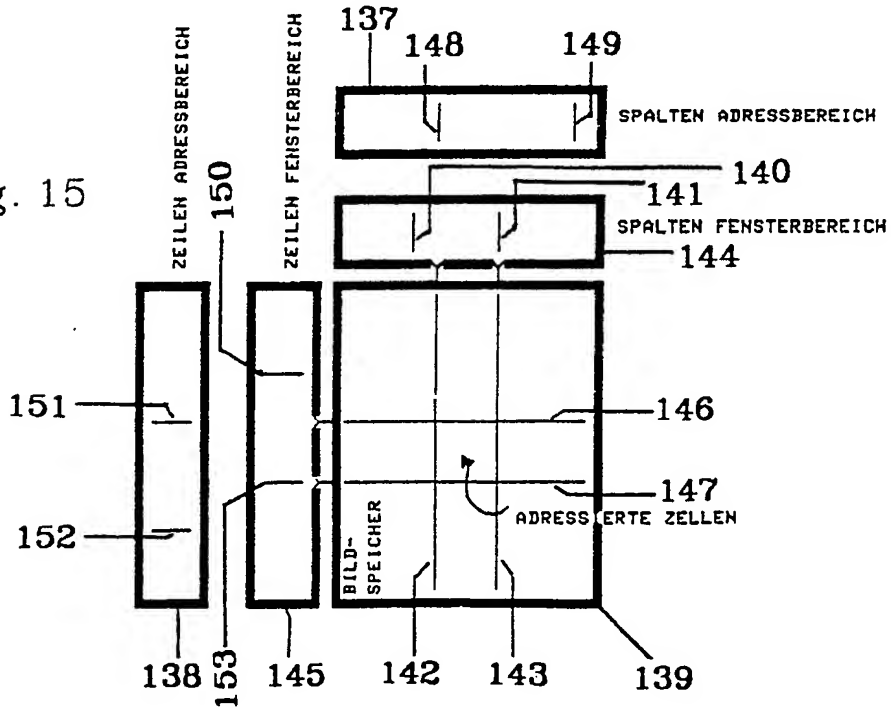
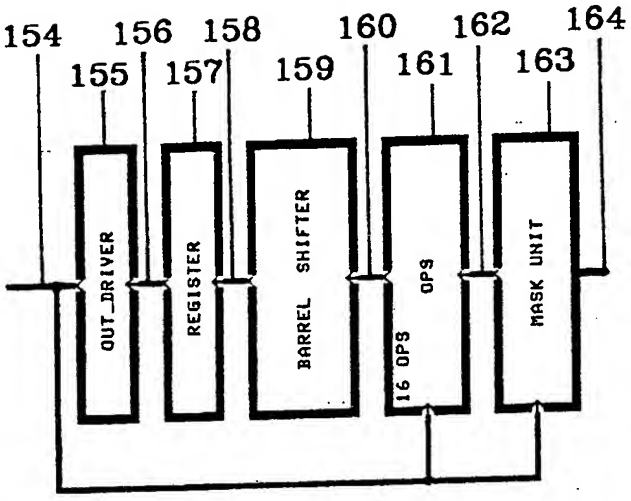


Fig. 16


154
155
156
157
158
159
160
161
162
163
164
OUT_DRIVER
REGISTER
BARREL SHIFTER
16 OPS
OPS
MASK UNIT

TRANSLATION OF DE 36 28 286A1

PATENT CLAIMS

1. The present invention relates to a processor in connection with a mapped memory, for example for a raster display, characterized by the fact that the processor is entirely or partially, and the mapped memory entirely, integrated on one chip.

2. Processor as per claim 1 is characterized by the fact that a silicon chip is used.

3. Processor according to claim 1 or 2 is characterized by the fact that the data-manipulating unit of the processor can - via a bus with a width corresponding to the number of memory cells in a row and/or column of the mapped memory - be coupled with the mapped memory.

4. Processor according to claim 1, 2 or 3 is characterized by the fact that the data-manipulating unit of the processor can consist of a barrel shifter with the bus width of the data-manipulating unit.

5. Mapped memory according to claim 1 or 2 is characterized by the fact that the memory field is equipped with a special connection logic which makes it possible to select and overwrite memory information stored in a virtual horizontal axis via a vertical axis. It is furthermore possible to select information stored in a virtual vertical axis via a horizontal axis.

6. Mapped memory according to claim 1, 2 or 5 is characterized by the fact that rows columns of memory cells - or parts thereof - can be read or written in a memory cycle.

7. Mapped memory according to claim 1, 2, 6 or 7 is characterized by the fact that rectangular blocks of memory cells, too, can be overwritten in an access cycle.

9. Mapped memory according to claim 1, 2, 6, 7 or 8 is characterized by the fact that the memory selection mechanism can consist of one or more M-of-N decoders.

10. Mapped memory according to claim 1, 2, 6, 7, 8 or 9 is characterized by the fact that the address lines selected by the address decoder can be changed with the help of, for example, a shift register.

11. Mapped memory according to claim 1, 2, 6, 7, 8, 9 or 10 is characterized by the fact that the memory sections can be blocked with the help of a logic as it is shown, for example, in figure 15.

12. Mapped memory according to claim 1, 2, 6, 7, 8, 9, 10 or 11 is characterized by the fact that the selection mechanism for the row of memory cells to be refreshed (refresh row) can consist of a shift register.

AL2

DESCRIPTION

The present invention relates to a processor in connection with a mapped memory, for example for a raster display, and is characterized by the fact that the processor and the image repeat memory are integrated on the same chip.

The present invention relates to a processor according to the overall definition in claim 1, especially for graphic applications.

The mapped memory content in a raster display system is to be modified quickly. For this, it is often necessary to modify several image pixels a the same time. This results in a fast image creation.

Conventional raster displays have a separate graphics processor and mapped memory. The mapped memory is, for example, connected to the graphics processor, by an 8, 16, or 32-bit databus. Such layouts carry the principal disadvantage that, in accordance with the width of the bus, only a limited number of image pixels can be modified in one graphics processing cycle.

The purpose of the present invention is the creation of extended access and modification means for the content of the mapped memory.

This task is solved through an application-matching layout of the identifying characteristics of claim 1.

The advantages that can be achieved with the present invention consist especially of the fact that rows and/or columns of image pixels or parts thereof can be shifted and/or modified in the mapped memory, and in the fact that rows, columns or blocks of image pixels can be overwritten. Particularly the generation of filled or hatched objects such as rectangle, parallelogram, trapez, circle, polygon, can be realized significantly faster than with means of current technical standards. Operations such as window clipping, scrolling and bit-block transfer (bit-bit) are possible with significantly increased speed. The generation of the synchronizing signals for the picture tube control can be integrated with little effort.

The present invention is described in detail with the help of drawings.

Figure 1 shows a conventional system.

Figure 2 - processor and mapped memory are connected by an expanded databus.

Figure 3 depicts the basic principal of the present invention.

Figure 4 shows the stylized memory field with identification of the databuses and drawn-in memory cells with two output buses.

Figure 5 shows a simplified memory cell.

Figure 6 depicts the memory field with the connections in the diagonal.

Figure 7 shows the principal memory layout.

Figure 8 shows the address path.

Figure 9 depicts the selection of a row of memory cells.

Figure 10 shows the selection of a column of memory cells.

Figure 11 shows the selection of a block of memory cells.

Figure 12 shows the three functional blocks of the decoder.

Figure 13 shows the address transformation unit.

Figure 14 shows a cell for the blocking of memory sections.

Figure 15 shows an example for the blocking of memory sections.

Figure 16 shows the data-manipulating unit.

A layout according to current technical standards is depicted in figure 1. Item 1 shows a processor. Item 2 is a secondary memory. Item 3 is a unit for the communication of the system with the external environment. Additional units 4 can be connected in parallel to the above. Item 7 is a image monitor controller unit which selects the image monitor information from memory 6 and depicts, through item 8, a picture tube. The parts 1, 2, 3, 4, 7 are, by a databus 5, coupled to a common memory 6. In this layout, it is the processor 1 that changes the image information in memory 6. The processor 1 and the memory 6 are located on different chips. The processor must modify the information in memory 6 via a databus with a width of 8, 16 or 32 bit. The communication speed between the units memory and processor is inhibited by item 5.

Figure 2 depicts a layout as per the present invention. The mapped memory 9 is connected with the processor 10 via the connecting lines located between the two. Both components 9 and 10 are integrated on one chip. A silicon chip is used in particular. Furthermore, the number of connecting lines will then be larger than those in layouts as per current technical standards. The number of lines is limited by the size of memory 9. It corresponds to the number of memory cells in horizontal and vertical direction in memory 9. Even if a part of the memory is not connected in this manner, the function will still be maintained for the remaining part. If additional, non-memory cells are coupled to the bus, the function will still be maintained. The processor can consist of 2 parts: a control unit and a data-manipulating unit. The control unit will not be considered further under this description, and may be located outside of the chip. The data-manipulating unit, hereinafter referred to as data path, and the memory are described as they relate to the present invention.

Figure 3 shows the principal connection between the mapped memory 14 with the address (item 11 and 12), refresh (item 32) and data paths (item 13), and their layout according to the present invention. Items 15 and 23 can be address decoders for the selection of memory sections in item 14. Items 17 and 25 can be units for modifying and/or storing of the address vectors generated in 15 and 23. Items 19 and 27 can be units for the blocking of memory sections. Items 20 and 29 can be units for the generation of memory refresh addresses. Items 20 and 29 can also be used for the multiplexing of the input vector from item 21 and/or 28. Item 32 can be a unit for supporting the display monitor refresh.

Item 39 can be a unit for avoidance of busing conflicts in 36. Item 40 can function as a register for storing a result. Item 41 can be a shift mechanism as performed by a barrel-shifter. Item 42 can be a logic processing unit which connects two input vectors with the help of logical operations. Item 43 can be a memory register which can store a vector and, in addition, facilitate the data exchange of this system with the external environment. The output of 15 can, via 16, be connected with the input of 17. The output of 17 can, via 18, be connected with the input of 19. The output of 19 can, via 20, be connected with the input for the column selection of item 14. The output of item 23 can, via 24, be connected with the input of 25. The output of 25 can, via 26, be connected with the input of 27. The output of 27 can, via 28, be connected with 29. The output of 29 can, via 30, be connected with the input for the row selection of item 14. Item 36 can be connected with items 14, 39, 43 and 42. The output of 43 can, via 38, be connected with the input of 42. Items 38 and 36 create the data inputs for item 42. These data inputs can transmit the vectors to be processed to 42. The output of 42 can, via 37, be coupled with the input of 41. The output of 41 can, via 35, be coupled with the input of 40. The output of 40 can, via 34, be coupled with the input of 39. Connections for the control of the components are not depicted.

Item 32 can be a unit for the support of the image display refresh. An output of item 14 can, via 31, be connected with the input of item 32.

The mapped memory (14) is located in the centre of the drawing (block mapped memory). The address paths 11 and 12 are connected above and left, the data path 13 right, and the refresh path 32 below with the memory. All connecting buses 16, 18, 21, 22, 24, 26, 28, 30, 31, 34, 35, 36, 37, 38 can, as the connecting width, have the number of rows in horizontal and vertical direction in the mapped memory.

The mapped memory can consist of a square field of memory cells. Each memory cell facilitates the storing of one image pixel. The field can be divided into two Part *A* can be depicted on the display monitor, in part *B* it is possible to store, for example, symbol sets or hatch patterns. The partition of the field in parts *A* and *B* can be selected at random since the field is constructed homogeneously. The partition can be determined during construction the of the chip.

According to the present invention, 12 and 11 can generate addresses which address, in 14, a memory section, for example a row or a column of memory cells, or parts thereof, or rectangular blocks of memory cells. In the addressed section can, via 36, information be written or selected. If information are selected, they can, via 36, be transmitted to 42 and 43. This information can now be stored in 43. In 42 this information can, via 36, be logically connected with an information from 43 and, via 37, transmitted to 41. In 41, this information can now be cyclically shifted bit by bit. This possibly shifted information can, via 35, be transmitted to 40, and intermittently stored there. If the information is not further transmitted from 14 to 36, 39 can, via 36, store the newly generated information in 14 or 42.

An information selected from 14 can, under the present invention, also be output via 31 to 32. Such is the case in a memory refresh cycle. Then, 20 transmits via 22 to 14, and 29 via 30 to 14 corresponding addresses, and 14 can, via 31, transmit display monitor information to 32.

Figure 4 shows, according to the present invention, the principal layout of the memory cells and the databuses of the memory. Item 44 and item 45 are databuses, via which the communication from and to the memory storage is realized. In this principal case, 6 data lines (items 46, 47) each are connected to each bus. Item 48 shows a memory cell. It is connected above and left with the data lines of the databuses 45 and 46. This facilitates select and write of rows of the memory cells by the bus "*A*" (44), and columns of the memory cells by the bus "*B*" (45).

Figure 5 shows a simplified static, dual-port memory cell to clarify the working mode of the memory storage. Items 50, 52 and 59 are transistors. 55 can, via the control line 54, facilitate row selection. 52 can, via the control line 54a, facilitate the column selection of the memory information input/output, via T2. 50 can, via the control line 51, facilitate the column selection of the memory information input/output via T1. The memory storage information is stored in the circuit, consisting of the two inverters 56 and 57 and the transistor 59.

The information can be selected when 52, 55 and 59 are open, or when 50, 55 and 59 are open. The information can be overwritten when 59 is closed and 52 and 55 are open, or when 59 is closed and 50 and 55 are open. Items 56, 57 and 59 can be replaced by a capacitor. In that case, the control line of 59 and the power supply lines of the cell are redundant.

According to the present invention, in all memory cells from figure 4, transistors 50 and 49 can be connected with the conducting databus lines of buses 45 connected to each corresponding cell. At the same time, all transistors 52 can, via 53, be connected to the databus line of the bus 44 connected to each corresponding cell.

This allows read and write transfer of data via the two buses 44 and 45. The transistor 55 facilitates the row selection, the transistor 50 facilitates the data exchange with bus "*B*", and the transistor 52 facilitates the data exchange with bus "*A*". In the memory field, all transistors 52 of one column of memory cells and all transistors 50 of one row of memory cells are connected with each other via the buses.

In order to furthermore select and/or write columns of memory cells through bus "*A*" and rows of memory cells through bus "*B*", additional connections can be established. Figure 6 shows the connections 60, 61, 62, 63, 64, 65, 66 and 67 which are prerequisites for this function in the diagonal of the memory field. Otherwise figure 6 corresponds to figure 4. By establishing these connections, each of the data lines of the bus "*A*" and the bus "*B*" are connected, which means:

Bus "*A*" line 0 is interconnected to bus "*B*" line 0 and
Bus "*A*" line 1 is interconnected to bus "*B*" line 1 etc.

The reading and writing of a row of memory cells via the bus "*A*", and the reading and writing of a column of memory cells via the bus "*B*", are now feasible. If, for example, a memory word is to be written into the first row via bus "*A*", the information residing on the bus can be read via the transistors 52 and 55 into the corresponding cells. If the left column of memory cells is to be overwritten via the bus "*A*", the data connection via the transistor 50 and 52 is to be selected in each corresponding memory cell. The second bit of the column will be set or deleted in the following manner:

The information on the second bit of the *A* bus 44 is transmitted to the connection point 61 at the *B* bus 45. From the *B* bus, the information can be loaded via the transistor 50 and 52 of the second cell into the cell. All these descriptions require that the area, which is to be read or modified, is addressed.

This address sector can be stored in 95 and in 97 or directly transmitted as an address to the mapped memory.

In 95 the stored address vector can be shifted or connected with an already stored vector. This facilitates very fast and versatile addressing possibilities. In the address transformation unit it is possible to chose between internal and decoder vector for the output vector. The address vector thus created will, in 97, be logically AND-connected with the vector residing there. This facilitates a locking of memory sections. The now created vector will, in the normal work mode, be given to the mapped memory as an access vector for the addresses. In the refresh cycle, a refresh vector is created instead of the aforesaid vector. This occurs in the last block of the address path.

Under the present invention, two (M-of-N) decoders 93 (one each for the row and column selection) can select a section of memory cells to be addressed. Each of these address decoders can be accessed with a largest or smallest desired address. The decoder then selects all memory rows or columns which are located between these two addresses. Through the two decoders it is now possible to address rows, columns or blocks of memory cells in the storage matrix.

Figure 9 depicts the selection of a row of memory cells. Items 101 and 102 can be, for example, two address decoders or address paths. Item 103 shows the memory storage. Item 104 shows the selected row in the memory. Item 105 shows the last selected row in the memory and item 106 shows the first selected column in the memory. Item 105b shows the address and memory section which may consist of several cells in a row.

Figure 10 shows the selection of a column of memory cells. The names of the units are identified in Figure 9. Item 107 shows the last addressed row, and item 108 the first addressed row in the memory. Item 109 shows the addressed column in the memory. Item 110 depicts the addressed memory section which may consist of several cells.

Figure 11 shows the selection of a block of memory cells. The names of the units are identified in Figure 9. Item 111 shows the first selected row, and item 112 the last selected row in the memory. Item 113 shows the first and item 114 the last selected column of memory cells. Item 115 shows the sector of memory cells which are addressed.

Figure 12 shows three functional blocks of the decoder. Item 117 shows a decoder which, from a set input address 119 on, releases all selection lines. The input address 119 can be coded, the selection lines 122 not. Item 118 shows a decoder which releases from an address 0 to a set input address 120 all selection lines 121. The input address 120 can be coded, the output address not. The following corresponding example is quoted below:

The buses 117 and 120 have a width of 9 bits, the buses 122 and 121 have a width of 512 bits.

The outputs of 117 and 118 are, via 121 and 122, connected with the inputs of 116. Here, both input buses can, bit by bit, be logically AND-connected with each other. The overall function resulting from this is as follows:

Through creation of a minimum address at 117 and creation of a maximum address at 118, a section of lines of the bus is selected at 123. The bus width of 123 corresponds to the bus widths of 121 or 122.

Figure 13 shows the address formation logic which facilitates the fast generation, storage and modification of address and address sections. Item 125 shows a shift register, which can be loaded and reloaded in parallel. This means, the stored vector in the shift register can be logically "OR"-connected with a vector available at 124 and stored. It is, furthermore, possible, to realize with these modified shift registers during the shift operation a logic "OR" or "AND" between the stored vector and the vector which has been shifted by one bit. This means for the logic "OR":

The vector 0001100000
becomes 0001110000
upon a shift to the right and 00111100000 upon a shift to the left. This means for the logic "AND":

The vector 0000111100
becomes 0000011100
upon a shift to the right and 0000111000
upon a shift to the left.

Item 127 is a multiplexer which can multiplex the buses 124 and 126. Item 129 is the same shift register as item 125. Item 131 is a logic unit which facilitates the connection of the two buses 128 and 130 with the help of logical operations. Item 124 is the input bus in 125 and 127. The output of 125 is, via 126, coupled with the input of 127. The output of 127 is, via the output of 129, coupled with the input of 131 via 131.

A bit vector can, via 124, be loaded into the shift register. Since the address decoder which generates the address inputs can generate only connected address sections, 125 or 129 serve as extended addressing means. Thus it is possible to address a random number of mapped memory sections which are not connected. Applications for this can be found if the same objects are drawn at the same time. After a first shift register a multiplexer can be located. With this multiplexer it is feasible to select the address vector from the shift register or the decoder.

The vector thus selected can be read into the second shift register. In the following block the address vectors of the second shift register and the address vector selected until that point can, with the help of the 16 possible logical operations, be interconnected. This chain of shift registers, multiplexers and logic units can be extended as needed. Items 124, 126, 128, 130 and 132 can have the same bus width as the outputs of the address decoders. Items 125, 127, 129 and 131 have the same processing width as the busses located in between.

According to the present invention, figure 14 depicts a one-bit-cell for blocking of memory sections. Item 136 is a latch. Item 133 is an input bit line and item 134 is the output bit line. Item 135 connects the output of 136 with 133 to 134 with the logic operation "AND". Upon setting the memory sector, which is to be protected from being overwritten, a 0 is written in each of the cells for blocking that row or column, and a 1 is written to enable each corresponding row or cell. The address path must contain such a cell for each address line in order to protect memory sections from being overwritten. This can occur in the clipping unit 19 or 27. Here, the address vector which is stored in that block is being logically "AND"-connected with the vector, resulting in the new address vector. If this procedure is applied to the address path of the row addresses and for the address path of the column addresses, a rectangular window can, for example, be written in the mapped memory. The remaining area of the mapped memory is then blocked.

Figure 15 shows an example. Items 137 and 138 can be M-of-N decoders. Items 144 and 145 can be the clipping unit for the row and column selection. Item 140 can be the smallest addressable column in the window. Item 141 can be the largest addressable column in the addressable window. Item 148 can be the smallest addressable column in the memory. Item 149 can be the largest addressable column in the memory. Item 150 can be the largest addressable row in the window. Item 153 can be the smallest addressable row in the window. Item 151 can be the largest addressable row in the memory. Item 152 can be the smallest addressed row in the memory. Items 142, 143, 146 and 147 abut the section in the memory which has now been addressed. The window, which was defined in 144 and 145, was smaller than the sector addressed in 137 and 138. By blocking certain sectors as described in this example a clipping, which is not based on mathematical operations, can be executed during the writing of a rectangle of memory cells. Thus the clipping costs only the time delay in 135.

In a dynamic semiconductor element, all memory cell contents must, during a time period depending on the generation process, be refreshed. Otherwise the information of the memory cell would be lost. In the depicted concept, all times of memory cells are sequentially refreshed. In a refresh cycle the column selection occurs in 20. The address decoder is disconnected, and all columns are selected. The row selection utilizes a dynamic shift register in 29 in which, after a reset phase, only one bit is set. The processing width of 20 and 29 corresponds to the width of their input and output buses. Once the addresses are stable, the memory can be enabled for read. This refreshes the addressed row of the memory cells. The display monitor refresh can be derived from the memory refresh. In each display refresh cycle the video register in block 32 can be filled with the part of data which are to be depicted on the display monitor. The memory sector, which is not to be shown, can for example be reserved for a set of symbols. This section can, with the help of block transfer operations, be transferred onto the visible part of the memory.

The content of 32 can be released in parts of, for example, eight bit each. Externally, these parts are, with the help of a fast shift register, converted into a serial bit stream. This bit stream controls the display monitor.

Figure 16 shows the data path. It is used for transformation and adjustment of graphic data from the mapped memory as well as for communication with the external environment. The width of the bus corresponds to the number of memory cells in a row or a column of the mapped memory. This facilitates the modification of very long vectors of the graphic pixel in an internal cycle. Item 164 depicts the data connection of the address path and the mapped memory to the external environment and the storage media for image pixel vectors in the data path. One vector can be loaded either from the image memory or word by word via the databus. The vector stored in 163 can, in the unit 161, be connected bit by bit with a vector which is selected from the display. All 16 logic operations between the two vectors are possible. The vector newly created by this operation can, for example in order to enable bit-bit (bit-Block-transfer), be rotated as needed by the barrel shifter. A register, located in the circuit after the shifter, facilitates the control of the data path. The subsequent tristate buffer functions to avoid busing conflicts in the I/O bus.

Comput. & Graphics Vol. 11, No. 2, pp. 185–192, 1987
Printed in Great Britain.

0097-8493/87 $3.00 + .00

Eurographics '86 Award Paper

THE INTEGRATED DISPLAY CONTROLLER (IDC) FOR VLSI-DESIGN WORKSTATIONS†

JÜRGEN STÄRK
Technische Hochschule Darmstadt, Fachgebiet Graphisch-Interaktive Systeme,
Alexanderstr. 24, 6100 Darmstadt, FRG

Abstract—A display controller to accelerate the updating of the frame buffer in raster displays is presented. The advantage of a frame buffer system is the ability to display pictures of any complexity, flicker free. The disadvantage is the great amount of pixel data, which must be updated, when the picture is changed. The Integrated Display Controller (IDC) described in this paper gets its name from the fact that both a pixel plane and a processor are integrated on a single chip. An internal logical unit, a shifter and special memory are provided to process complete rows and columns of pixels, or some part of them in parallel. All operations are performed in one bus cycle, which means that all the pixels addressed in the row or column are modified at the same time. Due to the special organization of the IDC, time consuming operations like block transfer or scrolling can be done significantly faster than in conventional systems. An experimental IDC with a lower screen resolution is developed to demonstrate the feasibility of the design. It is intended for use in VLSI-design workstations, where a great quantity of data has to be manipulated.

1. INTRODUCTION

Usually raster displays are based on the combination of two components:

- conventional dynamic memory circuits
- circuits used to periodically refresh the CRT from a frame buffer, circuits to refresh dynamic RAMs and circuits to scan-convert graphical primitives [1].

The main point of the effort in conventional design is to combine circuits with different properties in a form, such that the disadvantages stay within a tolerable range. These disadvantages become more evident with the increasing quality of the display (screen resolution and screen refresh rate). Therefore new approaches have to be found [2–5].

The features of microelectronics improve continuously. Higher integration densities are available every year. The 1 Mbit dynamic RAM is already available. Such an increase in chip complexity makes possible improvements of a higher functionality. Developments of a new raster display architecture should use this functionality in a new way to accelerate both image generation and image display speeds.

An analysis of the operations found in VLSI-design workstations revealed a performance bottle-neck in the drawing of vertical and horizontal lines, the filling of rectangles and in hatching. The problem presented itself in updating the video-RAM. Graphics processors access data from video memory using a 8- or 16-bit buses. This operation defines the time constraints. If all the objects to be drawn are loaded in the main memory, the generation of the image depends only on the graphics controller. Data access can be improved, if screen refresh can be operated through a second port of the RAM. But the speed-up factor is still not satisfactory. Another improvement can be made by an expansion of the memory bus. This step increases the performance, but also the demands of the electrical components. A solution of this problem is proposed in the design of the IDC.

2. SYSTEM OVERVIEW

The IDC contains the storage capacity and manipulation features for one bit plane. If color is required, several IDCs can be connected to work in parallel, one IDC for each bit plane. The IDC consists of 4 functional blocks shown in Fig. 1. The block GRAPHIC is described in detail below.

Figure 2 shows the arrangement of the block GRAPHIC on the chip. A special RAM is placed in the center. The address paths are connected at the top and to the left, the data path to the right and a refresh path at the bottom of the RAM.

3. MEMORY SYSTEM

The memory system consists of a special RAM and two address paths. These paths are composed of two m-of-n decoders, shift registers, latches and multiplexers. With this combination it is possible to read and write rows and columns of pixels and set blocks of pixels to any user defined pattern.

3.1. *RAM*

The RAM is the most important element of the IDC. Its architecture is the key to improved performance. The on-chip RAM is arranged in a rectangular array which provides space for display screen, and space for a character set. In order to simplify the design of a RAM cell, we preferred to use static cells in our first development, but the whole IDC is designed, ultimately, to work with dynamic RAM cells. In order to illustrate the way the RAM array works, a simplified dual port static memory cell is shown in Fig. 3.

It allows for read and write transfer of data in two directions. The pass transistor T1 serves for the selec-

† This paper is one of the award winning papers from EUROGRAPHICS '86 the annual Conference of the European Association for Computer Graphics. The paper is published here in a revised form, with permission of North Holland Publishing Company (Amsterdam), the Publisher of EUROGRAPHICS '86 Conference Proceedings.

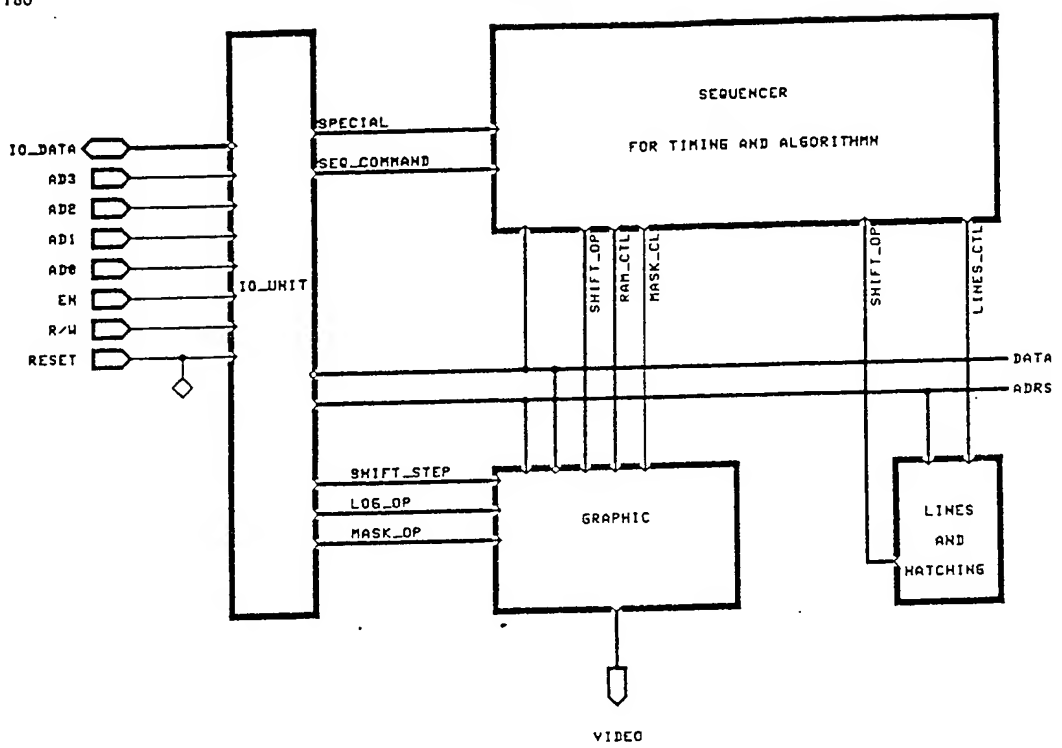


Fig. 1. The main parts of the IDC.

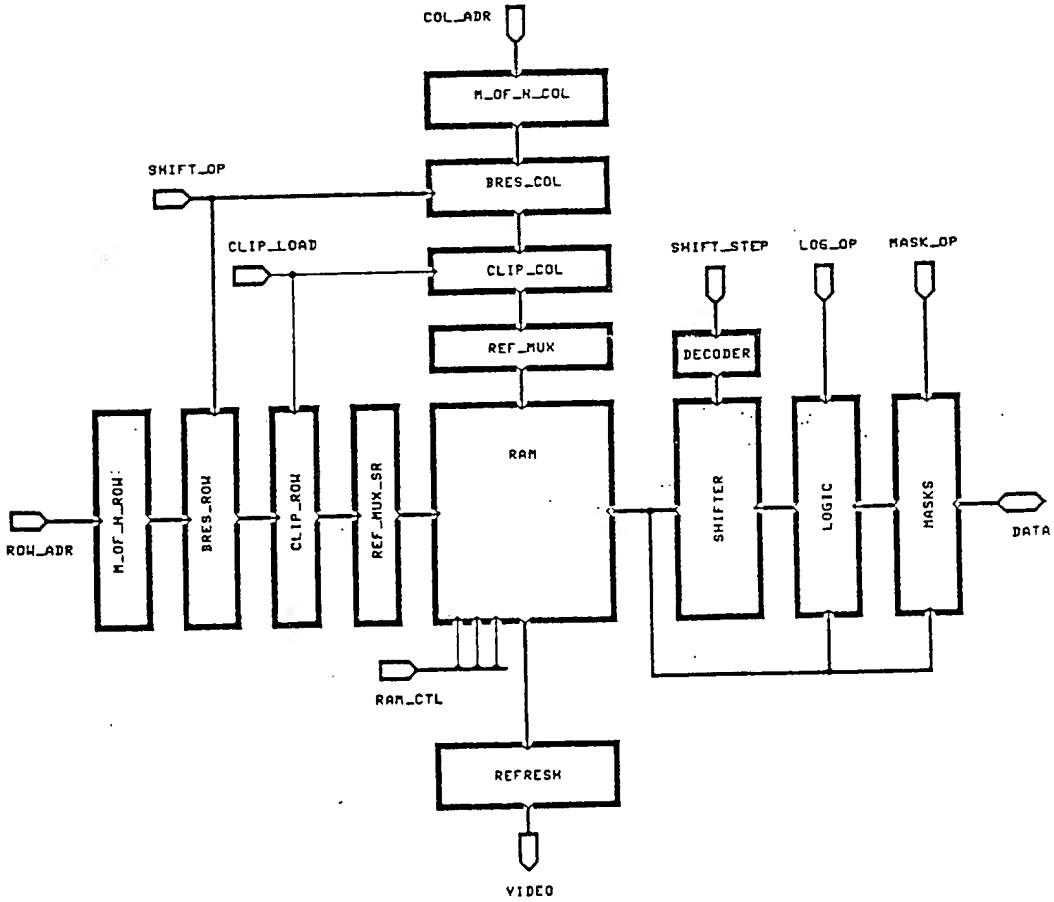


Fig. 2. The block GRAPHIC.

tion of the row; Transi
to the left and T3 to t
array all T2 and T3 t
row (wire IO2 in Fig
spectively (wire IO1 ir
ditional logic is neces
selection as shown in

In the first step of t
RAM is cleared. In the
into the first row. In t
read.

The functionality i
tions in the main diag
by 90 degrees, if row

3.2. *RAM decoders*

In a conventional
mechanism consists c
6). Only one row ca
which the required bi

In this design two
decoder is controlled
The decoder enables
Fig. 7).

With two decoder
selected (see Fig. 8–1

This mechanism e
To read data, only o
because parallel line
data connection.

3.3. *Window clippin*

In the IDC, windc
culations. It works
priate area of RAM
in block M_OF_N
it is possible to select
The areas can be st
and CLIP_COL to
lected and the storec
effective RAM addre

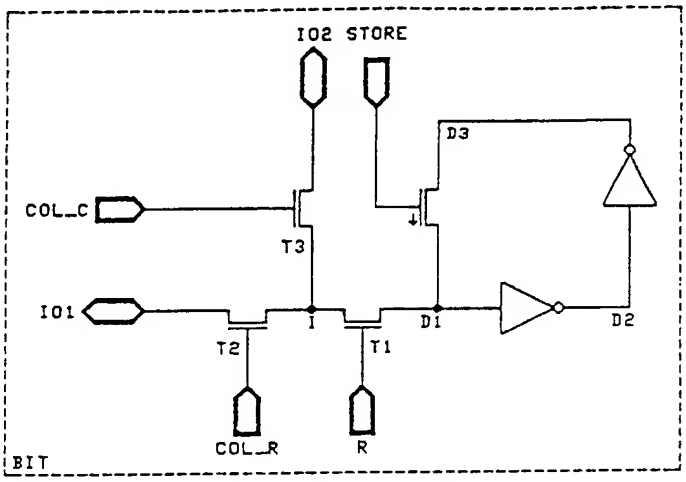


Fig. 3. A modified RAM cell.

tion of the row; Transistor T2 controls the flow of data to the left and T3 to the top of the cell. In the RAM array all T2 and T3 transistors are connected to the row (wire IO2 in Fig. 3) and column data lines respectively (wire IO1 in Fig. 3). A small amount of additional logic is necessary for the column/row output selection as shown in the top of Fig. 4.

In the first step of the example shown in Fig. 5, the RAM is cleared. In the second step, a pattern is written into the first row. In the third step, the first column is read.

The functionality is made possible by the connections in the main diagonal. This feature turns the data by 90 degrees, if row access is selected.

3.2. *RAM decoders*

In a conventional RAM the memory selection mechanism consists of a 1-of-n row decoder (see Fig. 6). Only one row can be selected at one time from which the required bit has to be chosen.

In this design two m-of-n decoder are used. Each decoder is controlled by a highest and a lowest address. The decoder enables all address lines in this range (see Fig. 7).

With two decoders an area of RAM cells can be selected (see Fig. 8–10).

This mechanism enables block selection for writing. To read data, only one row or column can be selected, because parallel lines of RAM cells use the same I/O data connection.

3.3. *Window clipping mechanism*

In the IDC, window clipping doesn't need any calculations. It works by enabling/disabling the appropriate area of RAM cells. With two m-of-n decoders in block M_OF_N_ROW and M_OF_N_COL, it is possible to select areas between two given addresses. The areas can be stored in the blocks CLIP_ROW and CLIP_COL to define the window. Later, the selected and the stored areas are combined to form the effective RAM address (see Fig. 11).

3.4. *Refresh logic*

In a dynamic RAM array every cell must be refreshed during a time that depends on the manufacturers process (~2 ms). Otherwise the information in the cell is lost.

Simply, refresh means: read the contents of a RAM cell into a buffer and write the value back into the same cell. Then the charge in that cell is refreshed.

The memory refresh in the version presented here is prepared for the future use of dynamic RAM cells. In this development it only supports the screen refresh.

In the IDC all rows of RAM cells are sequentially refreshed. In a refresh cycle the column selection is done in block REF_MUX in Fig. 2. The RAM decoder will be switched off and all columns will be selected. The row selection uses a dynamic shift register and is done in block ROW_REF_SR in Fig. 2. When the addresses are stable the RAM can be enabled to be read. The data from the addressed row of cells are buffered in a refresh buffer contained in block REFRESH in Fig. 2 at first, so that it can be used to refresh dynamic RAM. The screen refresh is derived from the memory refresh. Every screen refresh cycle the video register contained in block REFRESH in Fig. 2 is filled with that part of the data from the refresh buffer which will be displayed on the screen. The memory not used for displaying is reserved for a user defined character set. This part can also be copied onto the visible area of the RAM using bitblt (see Fig. 12).

The output of the video register is divided into eight-bit parts. Externally these parts are transformed into a serial bit stream by a fast TTL-shift register to provide data for the TV in the usual way.

4. DATA UNIT

The data unit is used for transforming and justifying the pixel data in the pixel store. The bus width in this unit is the same as the number of RAM cells in a row or column in the pixel plane, so it is possible to modify a very long pixel vector in one internal cycle. The unit

AL2

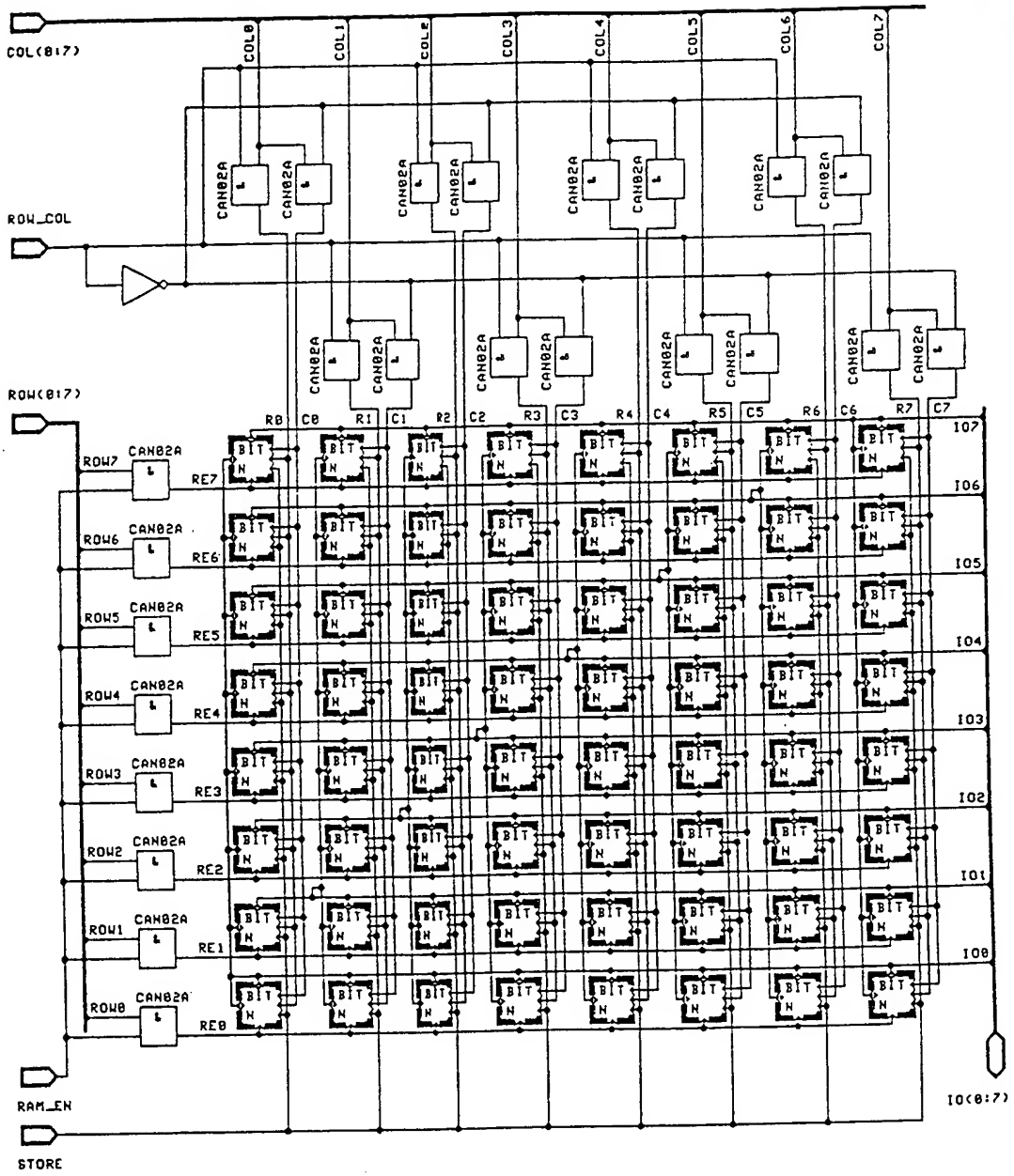


Fig. 4. The RAM array.

		Step 1	Step 2	Step 3
		COLUMNS 1 4		
R	1	0000	1010	1010
O		0000	0000	0000
W		0000	0000	0000
S	4	0000	0000	0000
		0000	1010	1000
		CLEAR BLOCK	WRITE ROW	READ COLUMN

Fig. 5. Clear, write and read cycle of operations on a 4 × 4 bit RAM array.

consists of the blocks MASKS. LOGIC and SHIFTER shown in Fig. 2.

The block MASKS represents a unit, which controls one source operand of the logical unit in the block LOGIC. In MASKS it is possible to select one of the following bit pattern:

- all bits one
- all bits zero
- a user defined pattern, which was read in sequentially through the DATA bus (see Fig. 2)
- a mask, which was preloaded from the pixel store

The block LOGIC consists of a simple logical unit with the boolean operations AND, OR, XOR, NOP

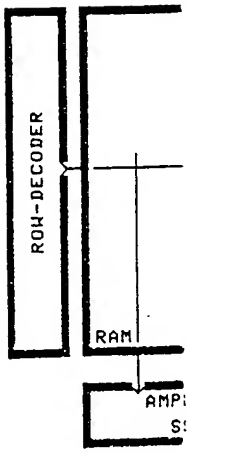


Fig. 6. Convention

and their negation. Two i
mask unit, the other from
bined logically in one cycl
unit is the input of the shift
SHIFTER in Fig. 2. The
writing back to the RAM.

5. GRAPHIC

While generating a pict
functions such as "line dr
speed up the generation of
ics functions must be integ
IDC supports some low l
powerful bitblt operations

5.1. *Line drawing*

In the IDC we distingu
and columns and arbitrary
be addressed directly and
The pixel vector will be
combined with a pattern
cessed area must be set to
in the mask unit, it can b

To draw arbitrary lines
BRES_COL in Fig. 2 are
of shift registers, which ca
from the address decoder
row, a column or a block
this the shift registers ar
generated by the address
gorithmn (Bresenham) ge
width by controlling the
This mechanism has thre

ADDRESS LIN	
MIN (0:2)	MA
0 0 1	0
0 1 1	1
0 0 0	1

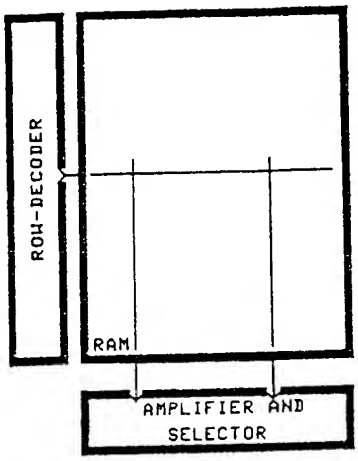


Fig. 6. Conventional RAM selection.

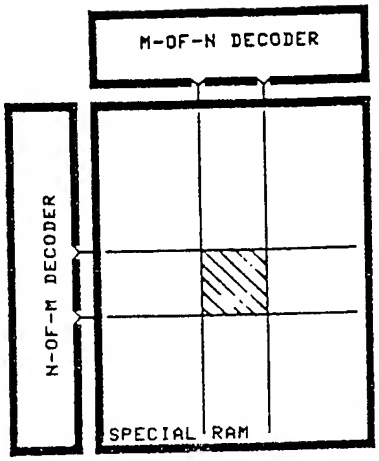


Fig. 8. A rectangle is selected in the RAM of the IDC.

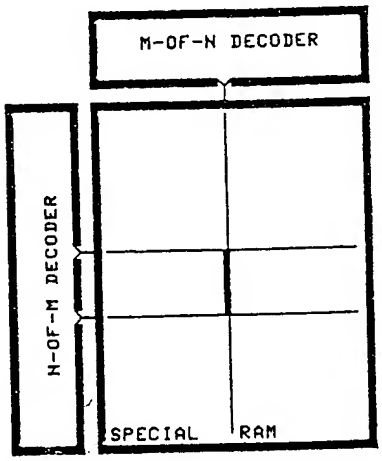


Fig. 9. A column is selected in the RAM of the IDC.

and their negation. Two input vectors, one from the mask unit, the other from the pixel plane can be combined logically in one cycle. The output of the logical unit is the input of the shifter, represented by the block SHIFTER in Fig. 2. The shifter justifys the data for writing back to the RAM.

5. GRAPHIC FUNCTIONS

While generating a picture on the screen graphical functions such as "line drawing" are performed. To speed up the generation of the picture low-level graphics functions must be integrated in the hardware. The IDC supports some low level functions and also the powerful bitblt operations.

5.1. *Line drawing*

In the IDC we distinguish between drawing rows and columns and arbitrary lines. A row or column can be addressed directly and modified in one clock cycle. The pixel vector will be read out of the pixel plane, combined with a pattern and written back. If the accessed area must be set to a predefined value contained in the mask unit, it can be set directly.

To draw arbitrary lines the blocks BRES_ROW and BRES_COL in Fig. 2 are provided. The blocks consist of shift registers, which can be loaded with the output from the address decoders. To draw a line: a point, a row, a column or a block must be accessed. To archive this the shift registers are loaded with the addresses generated by the address decoders. A line drawing algorithmn (Bresenham) generates a line of the required width by controlling the shift registers and the RAM. This mechanism has three advantages:

- The following types of lines can be generated very fast (see Fig. 13).
- The shift register needs less time to generate the new addressed area.
- The hardware for the Bresenham algorithmn becomes smaller.

5.2. *Rectangular block drawing*

In the IDC we must distinguish between writing and modifying a block of pixels. To set a block of pixels we only have to select the pattern in the mask unit and set the block addresses. Then the IDC fills that area with the given pattern in one internal cycle.

To modify a block, every row or column must be

ADDRESS LINES		RAM SELECT LINES							
MIN (0:2)	MAX (0:2)	SEL0	SEL1	SEL2	SEL3	SEL4	SEL5	SEL6	SEL7
0 0 1	0 1 1	0	1	1	1	0	0	0	0
0 1 1	1 1 1	0	0	0	1	1	1	1	1
0 0 0	1 1 1	1	1	1	1	1	1	1	1

Fig. 7. Example of the RAM selection.

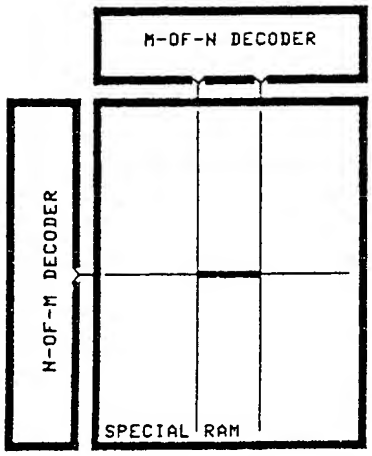


Fig. 10. A row is selected in the RAM of the IDC.

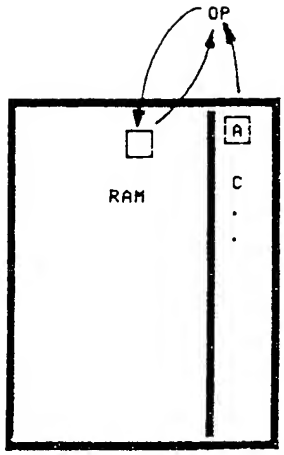


Fig. 12. Bitblt done with the IDC.

read out of the pixel memory sequentially. Then it can be modified and written back.

5.3. *Stroke precision characters and brush drawing*

The shift register can also be used to draw stroke precision characters or to perform brush drawing on the screen. Only the first point, row, column or block must be set. Then two shift registers can be loaded (BRES_ROW and BRES_COL in Fig. 2) which act as pointers to the pixel plane. These pointers can be positioned with simple commands like up or down. In a future design it is planned to have an additional RAM on-chip to store vectors for a character set.

5.4. *Scrolling*

The IDC supports scrolling up, down, right or left in a given window. Scrolling is performed by reading a row or a column, shifting it in the data unit and writing it back. A sequencer controls the timing and the sequential access of rows or columns. If a window is selected, the scrolling is executed only in the window.

5.5. *Hatching*

Hatching is controlled by the sequencer and the hardware for line drawing. First the pattern for hatching must be loaded in the block MASKS (see Fig. 2) in a register. Then the logical operation for the hatching must be enabled and the shift steps declared. A rectangular area or one of the line types (see Fig. 13) must be selected. The hatching is started by writing a control word into the IDC, which enables the sequencer and the hardware for line drawing. A row or a column of RAM cells will be read out, modified in the data unit and then written back. Then the mask must be moved in the shifter to generate the new mask for the next hatch cycle.

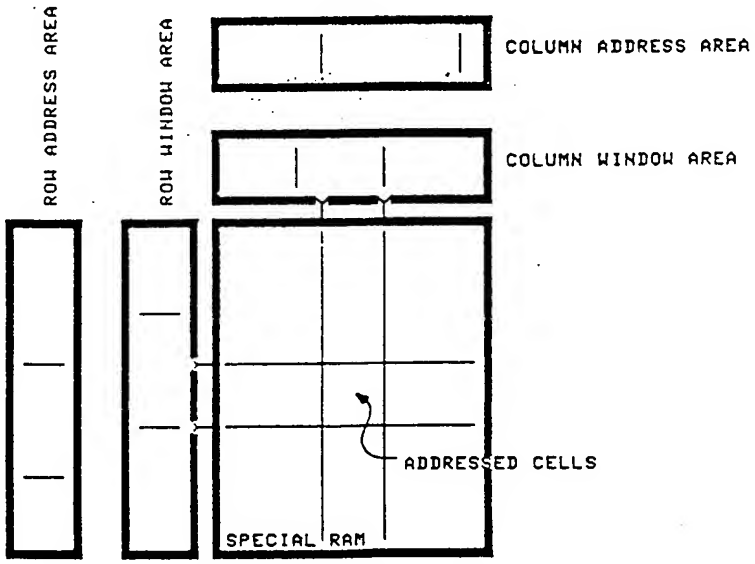


Fig. 11. Window clipping during a block write in the IDC.

5.6. *Bitblt*

Bitblt are powerful c
arrays. A source area v
ation area by a logical
placed in the destinatic

The IDC supports b
To start a bitblt the da
must be initialized, the
into the shift regist
BRES_ROW shown i
dresses into the addres
can be started. A row
read out of the pixel
data to the destination
register contained in b
row or column of the
the RAM, logically co
and then written back

Table

ops	relati
1	point: RMW row/column: RMW · n block: RMW · n · m
2	row/column: block:
3	n · RMW · CC
4	n · RMW · CC
5	$n \cdot \frac{m}{8} \cdot RW \cdot C$
6	n · RMW · CC
7	n · RMW · CC

m, n	
RMW	(6 cycles)
WC	(1 cycle)
RC	(1 cycle)
RW	(2 cycles)
ST	(1 cycle)
PRE	(1 cycle)
ICT	(150 ns)
COM cycle	(10 cycles)

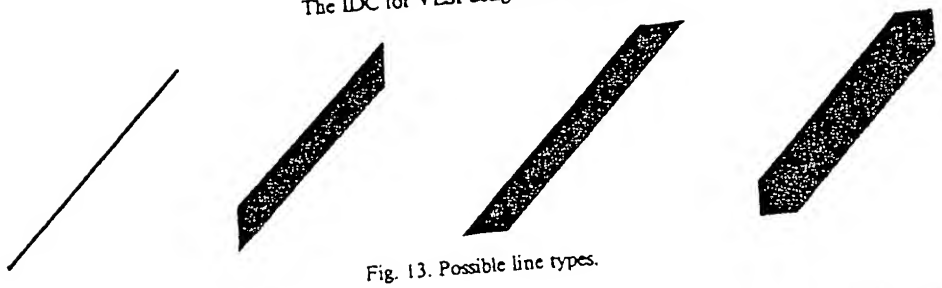


Fig. 13. Possible line types.

5.6. *Bitblt*

Bitblt are powerful operations to combine two pixel arrays. A source area will be combined with a destination area by a logical operation and the result will be placed in the destination area.

The IDC supports bitblt controlled by a sequencer. To start a bitblt the data transfer unit and the shifter must be initialized, the source address must be loaded into the shift registers (block BRES_COL and BRES_ROW shown in Fig. 2) and the destination addresses into the address registers. Now the sequencer can be started. A row or a column of pixels will be read out of the pixel memory, shifted to justify the data to the destination area and loaded into the mask register contained in block MASK in Fig. 2. Then a row or column of the destination area is read out of the RAM, logically combined with the source vector, and then written back into the RAM. Now the source and destination addresses are incremented. These steps are repeated until the whole block has been worked through.

6. PERFORMANCE

6.1. *Comparison*

In Table 1 the following comparison is made between the IDC and a conventional monochrome raster display. For all calculations an average line or block length of 100 pixels is assumed.

Operations of Type 1 are simple read, write or modify operations.

Type 2 is like type 1, but an external mask must be loaded.

Type 3 is line drawing with a given width. In the IDC the line drawing can be used as block fill of a parallelogram. We assume a width of 10 pixels. For

Table 1. Comparison between the IDC and a conventional monochrome raster display

	conventional		IDC	
ops	relative	absolute	relative	absolute
1	point: RMW	6 cycles	$2 \cdot WC + ST$	3 cycles
	row/column: RMW · n	$6 \cdot 10^2$ cycles	$4 \cdot WC + ST$	5 cycles
	block: RMW · n · m	$6 \cdot 10^4$ cycles	$4 \cdot WC + ST$	5 cycles
2	row/column:		$4 \cdot WC + \frac{n}{8} \cdot PRE + 2 \cdot ST$	19 cycles
	block:		$4 \cdot WC + n \cdot \left(\frac{m}{8} \cdot PRE + 2 \cdot ST\right)$	$1.5 \cdot 10^3$ cycles
3	n · RMW · COM · width	$6 \cdot 10^4$ cycles	$6 \cdot WC + 2 \cdot ST + n \cdot ICT$	8 cycles + 10^2 ICT
4	n · RMW · COM · width	$6 \cdot 10^4$ cycles	$6 \cdot WC + 2 \cdot ST + n \cdot ICT$	8 cycles + 10^2 ICT
5	$n \cdot \frac{m}{8} \cdot RW \cdot COM$	$2.6 \cdot 10^3$ cycles	$4 \cdot WC + ST + 2 \cdot ICT \cdot n$	5 cycles + 200 ICT
6	n · RMW · COM · width	$6 \cdot 10^4$ cycles	$6 \cdot WC + 2 \cdot ST + n \cdot ICT$	8 cycles + 10^2 ICT
7	n · RMW · COM · width	$6 \cdot 10^4$ cycles	$6 \cdot WC + 2 \cdot ST + n \cdot (ICT + 1)$	8 cycles + 10^2 ICT

m, n		size of pixels.
RMW	(6 cycles)	computing the memory address, read cycle, bit justify, modify the data, again justify and write back the data.
WC	(1 cycle)	write cycle.
RC	(1 cycle)	read cycle.
RW	(2 cycles)	read and write cycle.
ST	(1 cycle)	write cycle for commands to the IDC.
PRE	(1 cycle)	preload time for a special mask (8 bit) in the IDC.
ICT	(150 ns)	internal cycle time.
COM	(10 cycles)	time to compute the next point.
cycle		a bus cycle time of the workstation processor bus.

the IDC there is no difference between drawing a narrow or a wide line.

Type 4 is brush drawing and stroke precision characters. Due to the fact that the internal control of the stroke precision character generation is not yet developed, stroke generation has to be controlled external. It requires the same time as brush drawing.

Type 5 is scrolling in a given window.

Type 6 is hatching.

Type 7 is bitblt.

6.2. *Window clipping*

Working with pixels, rows, columns or blocks, the window clipping is done concurrently by the hardware. If we use graphical operations, which are performed internally sequentially, we must distinguish between drawing in small or large windows (see Fig. 14). In such cases the IDC tries to draw all points inside and outside any window, so the time to perform the algorithm depends on the total number of internal steps and not on the size of the window.

6.3. *Additional performance of an IDC system*

The calculation of pixel addresses of the screen coordinates is not necessary any more. The chip accepts column and row addresses directly. Also there is no need to adjust the data within a word to preserve word boundaries.

7. CONCLUSION

Low cost, high performance raster graphics requires the development of inexpensive and highly parallel raster architectures. The IDC achieves these goals by placing the parallelism inside a chip.

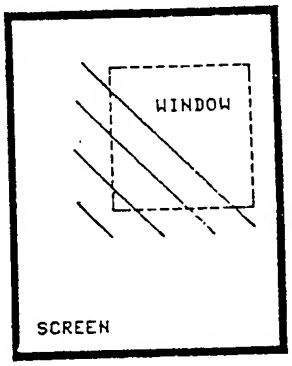


Fig. 14. Small and large lines in windows.

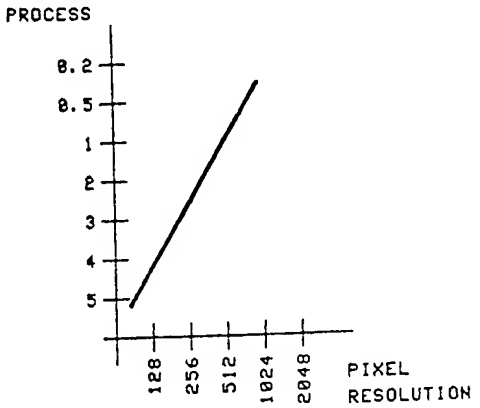


Fig. 15. Relation between available process and the number of RAM cells in the IDC.

The architecture of the IDC is not fixed by the presently available integration densities in VLSI. On one hand as VLSI densities go up so the design can be increased in size to give higher resolution stores, on the other as the resolution is increased so the performance of the unit will be improved. Figure 15 shows the relation between the number of RAM cells (and therefore the screen resolution) and the available process. The first IDC is designed for a low resolution application, because of the given process parameters and the use of static RAM cells.

Acknowledgements—Many thanks to Prof. Dr. W. Strasser for his guidance in this project and the discussions about raster graphics, to F. J. Grosch, W. Kandek, Th. Moissiadis and G. Sakas for their help in simulation and full custom design and also to W. Bonat, H. Joeppen and N. Wehn for their support with VLSI-design software. This work has been supported by the German Government, project E.I.S. (development of integrated circuits).

REFERENCES

1. W. Newman and R. F. Sproull. *Principles of Interactive Computer Graphics*, second edition. McGraw-Hill (1979).
2. J. H. Clark. Distributed processing in a high-performance smart image memory. *Lambda*.1, 40–45 (1980).
3. S. Demetrescu. High speed image rasterization using scan line access memories, in: 1985 *Chapel Hill Conference on VLSI*. Computer Science Press, Rockville. Maryland (1985).
4. S. Gupta. R. Sproull and I. E. Sutherland, A VLSI architecture for updating raster-scan displays, in: *SIGGRAPH Conference Proceedings*, pp. 71–78. Dallas (1981).
5. D. S. Whelan, A rectangular area filling display system architecture, *Computer Graphics* 16, 147–153 (1982).

Comput. & Graphics Vol. 11, No. 2, pp.
Printed in Great Britain.

CROSS-SE

Laborat

Abstract—A theory
surface construction
means of interactior

1. INTRC

In modeling free-form sh
B-splines [1] have becom
surface representation. Th
the cubic *B*-splines is clea
uous representations. En
β-splines [2], the modeli
B-splines may be further
B-splines one can represe
spheres and tori precisely

While the theory of t
been thoroughly surveye
emphasis has been put o
face construction in pra
ulation and verification
dimensional display invo
for an experienced CAI
spline representation doe
surface locations, but or
The initial definition for
carried out by trial and
adjusting the surface sh
Rather then dealing with
surface construction an
directly to the surfaces s

With the principles of
surface modeling task c
curve analysis. To comb
face description, the me
apply for the most gene
eration can be used for
surfaces. Surface inter
data can similarly be
methods. The cross-se
completed with the int
metric curves on the *B*-

These methods buil
splinES) free-form surfa
at our laboratory. The
AGX solid modeling s
tation [5], using Frank
The implementation o

† This paper is one of t
ROGRAPHICS '86 the a
Association for Compute
here in a revised form.
Publishing Company (A
ROGRAPHICS '86 Conf